Apple Answer Book

Index

July 3, 1980

"CANNED" LETTERS

A. APPLESOFT

[01] ONERR GOTO info
[02] DIM A$(N) is N+1 strings
[03] OUT OF MEMORY error
[04] Real number precision
[05] Random number generator isn't random (patterns)
[06] Round FP numbers & set decimals
[07] "PRINT USING" simulator
[08] Relocate Applesoft program
[09] Integer to Applesoft program conversion
[10] Tones don't work
[11] Use FRE(0) for garbage collection
[12] FRE(0) gives a negative number
[13] Zero page use and pointers

B. Pascal

[01] Pascal with single disk
[02] Sending things to the modem
[03] Sending things to the printer
[04] Difference between REPEAT..UNTIL & DO..WHILE
[05] Non-Apple Interface Card
[06] System configuration to run Pascal
[07] Terminal setup
[08] Intrinsic Units
[09] Include file
[10] Compile Swapping mode ($S+)
[11] Copy BASICS disk
[12] Extra linefeeds on printer
[13] Segment Procedures
[14] Compile with SYSTEM.LIST no code made ($L+)
[15] Amount of free run-time memory
[16] Debugger
[17] Save HIRES screen to disk
[18] Filer transfer to PRINTER:
[19] Micro-Modem support
[20] Long Integer fix
[21] Real number format
[22] Pascal exponents
[23] TTLOUT problem
[24] No CHAIN in PASCAL
[25] Capital letters required
[26] NOT problems
[27] SYSTEM.CHARSET
[28] CLEARSCREEN
[29] CALC
[30] INTRINSIC DATA SEGMENT
[31] Terminal Driver package

C. DOS

[01] How to BLOAD/BSAVE
[02] Printer-DOS disconnect
[03] Crashed disk recovery
[04] Software slot finder
[05] 6502 interfacing
[06] BLOAD A,L parameters
[07] Direct vs Indirect commands
[08] APPEND overwrites start of file
[09] Reset re-boots the disk
[10] DOS commands and GET

D. MINI-ASSEMBLER

[01] How to use

E. APPLE II PLUS

[01] Mini Assembler not there
[02] How to run an Integer Basic program
[03] APPLE II vs APPLE II Plus

F. MONITOR ROM

[01] S and T commands in Autostart ROM
[02] Accessing the monitor
[03] Autostart ROM reset vector

G. FIRMWARE CARD

[01] What the switch does
[02] Contact oxidation

H. LANGUAGE CARD

[01] Do you get both Basics with language card?
[02] Microchess doesn't work
[03] Programmers Aid #1 ?

I. CASSETTE INTERFACE

[01] Cassette routines
[02] Tape loading problems

J. PRINTER INTERFACING

[01] Controlling a printer from within a Basic program
[02] Centronics 730 normal/expanded printing
[03] Tabbing on printer
[04] 6502 printing
[05] Centronics 779 gives extra linefeeds
[06] Centronics 779 gives no linefeeds

K. SERIAL CARD

[01] Printer handshaking
[02] Line width > 40 trashes program (Applesoft)

L. COMMUNICATIONS CARD

[01] Download from mainframe to text file
[02] SOROC & Applesoft, no linefeeds
[03] Modify ACIA status
[04] Datamover problem
[05] 4800 baud missing wire
[06] Print routine fix for 3.2 or 3.2.1 DOS
[07] Communications Card handshake
[08] <CTRL-R> for remote terminal
[09] How to use with modem
[10] 9600 baud modifications

M. PARALLEL PRINTER CARD

[01] Configuration block

N. GENERAL INTERFACING

[01] 20 ma current loop
[02] Interface to TV station
[03] TTY driver
[04] Card reader sources
[05] Numeric keypad

O. HARDWARE

[01] Eurapple conversion
[02] Apple power consumption
[03] Non-Apple memory expansion-speed
[04] Interrupts

P. VIDEO

[01] 80 chars per line on Apple screen

Q. HIRES

[01] Character sets
[02] Shape table BSAVE & BLOAD
[03] Screen mapping & colors
[04] HGR trashes RAM Applesoft
[05] Plotting on both screens

R. APPLEPOST

[01] Non-Apple printer card
[02] Used the sample program

S. RENUMBER

[01] Renumbers after a "*"

T. MISCELLANEOUS

[01] RAM test of top 16K with DOS
[02] Back issues of CONTACT
[03] PEEKs, POKEs, and CALLs
[04] Availability of source code and schematics

Applesoft

A. APPLESOFT

[01] ONERR GOTO info

If you are using ONERR GOTO, please carefully read pages 81, 82, and 136 of the Applesoft Reference manual. There are two correct ways to leave your ONERR routine. First, you can use RESUME and Applesoft will re-execute the statement that caused the error. Or you may use the machine language routine on pages 82 and 136 (page 136 is the better example) and a GOTO. Your program need only POKE this routine into memory once. It will not affect the level of GOSUBs or FOR...NEXT loops in the program.

[02] DIM A$(N) is N+1 strings

Applesoft supports multi-dimensional string and numeric arrays. When Applesoft executes the statement "DIM A$(5,100)" it actually sets aside 606 elements. That's because Applesoft uses element number 0, so it creates an array of 6 by 101 elements (5+1 by 100+1). Please see page 58 of the Applesoft Reference Manual for more information.

[03] OUT OF MEMORY error

Enclosed is information concerning OUT OF MEMORY ERRORS in Applesoft.

(Please see attached App. Note #B11)

[04] Real number precision

Applesoft stores its real numbers in a binary floating point notation. Since the mantissa is 32 bits long, Applesoft has 9 significant digit precision. Please note that some of the mathematical functions lose accuracy near their boundary conditions.

[05] Random number generator isn't random (patterns)

Enclosed is a copy of an applications note describing Applesoft's random number generator.

(Please see attached App. Note #B10)

[06] Round FP numbers & set decimals

There is a formula to round Applesoft real numbers on page 18 of the Applesoft Reference manual and a program to limit the number of digits to the right of the decimal point on page 22. In addition I am including a listing of a program to format real number output.

(Please see attached App. Note #B8)

[07] "PRINT USING" simulator

Applesoft doesn't contain a PRINT USING statement. However I am inculding a listing of a program that aligns decimal points in a printout.

(Please see attached App. Note #B8)

[08] Relocate Applesoft program

The easiest way to use HIRES graphics with a long Applesoft program is to relocate the program. Do the following before loading your program or as a seperate loader program:

For HGR	For HGR and HGR2
POKE 103,1	POKE 103,1
POKE 104,64	POKE 104,96
POKE 16384,0	POKE 24576,0
LOAD (your program)	LOAD (your program)

These instructions protect HIRES memory by starting the Applesoft program above it.

[09] Integer to Applesoft program conversion

Page 76 of the DOS 3.2 Reference Manual contains a routine which can be used to convert Integer Basic programs to Applesoft using the disk system. Integer Basic is required for the first step of the conversion only.

First, load in the Integer program. Add the routine on page 76 at the very beginning of the program, adjust the LIST line for the specific program, and give the textfile a unique name. Then type "RUN". This will cause the program to be listed out to the disk as a textfile under the name specified. At this point, the text file is independent of the language.

To convert, type "EXEC <your filename>" in Applesoft. The textfile will be read into the system as if typed from the keyboard. Since Applesoft does not check syntax at input time, the Integer syntax is accepted, although the program may not run. Save the program in Applesoft as usual.

Micro Magazine, January 1980, contains an article describing Integer to Applesoft conversion using cassette instead of disk. The article also describes some of the syntax and logic changes that will usually have to be made.

[10] Tones don't work

The tones subroutine in the old Apple II Reference manual (the Red Book) will not work with Applesoft because it uses memory needed by Applesoft. I am including a copy of the tones subroutine that has been relocated to a safe area in memory.

(Please see attached App. Note #B5)

[11] Use FRE(0) for garbage collection

Applesoft stores its string variables dynamically. However when it runs out of free memory it must clean up the unused strings, a process which can take several minutes. There is no way to stop this process but you can give your program's user some warninig. The following line should be put in the main flow of your program.

```
1000 IF PEEK(112)-PEEK(110)<4 THEN PRINT"STANDBY": A=FRE(0)
```

[12] FRE(0) gives a negative number

Sometimes PRINT FRE(0) prints a negative number. This will only happen in a 48K Apple and is caused by the way Applesoft handles integer numbers. Any number greater than 32767 will appear to have 65536 subtracted from it. So just add 65536 to the value returned. This can be done from a program like this:

```
1000 PRINT FRE(0) - 65536 * (FRE(0) <0)
```

[13] Zero Page use & pointers

Applesoft's zero page usage is documented in the Applesoft Reference manual on pages 140 and 141. There is further documentation in the new Apple II Reference manual on page 74.

Pascal

B. Pascal

[01] Pascal with single disk

On a single drive PASCAL system, each major function (editor/filer, compiler/assembler, etc.) should have a "stand-alone" diskette, consisting of everything necessary to allow the system to function without swapping diskettes. For example:

A stand-alone editor system could be configured as:

```
SYSTEM.PASCAL
SYSTEM.MISCINFO
SYSTEM.EDITOR
SYSTEM.FILER
<plus text files to be edited>
```

A stand-alone compiler system could be configured as:

```
SYSTEM.PASCAL
SYSTEM.MISCINFO
SYSTEM.LIBRARY
SYSTEM.COMPILER
SYSTEM.LINKER
SYSTEM.EDITOR (optional for debugging)
SYSTEM.SYNTAX (also optional)
<plus the file to be compiled>
```

...and so on.

NOTE: SYSTEM.APPLE does not have to be placed on the disk since it is only loaded during the initial boot; however, by making it resident, each disk would be fully capable of a cold start (power-up or reset).

[02] Sending things to the modem

Please see the note below. Use the same program except replace "PRINTER:" with "REMOUT:".

[03] Sending things to the printer

Data can be directed to other system devices in a PASCAL program with the addition of a few statements. for example, let's print a string on the printer:

```
PROGRAM DEMO;

VAR REPORT:TEXT;
    ANSWER:STRING;

BEGIN
  WRITE('STRING TO PRINT : ');
  READLN(ANSWER);
  REWRITE(REPORT,'PRINTER:');
  WRITELN(REPORT,ANSWER)
END.
```

Now, to change devices, simply change the "PRINTER:" in the rewrite statement to the name of the device to write to. For an example of writing to a disk file, see APPLE3:DISKIO.

[04] Difference between REPEAT..UNTIL & DO..WHILE

PASCAL supports two forms of conditional loop: REPEAT..UNTIL and WHILE..DO. A loop of the form

```
REPEAT
.
.
UNTIL <condition>
```

will be executed at least once, even though the condition is satisfied when the loop is initially entered.

The loop consisting of

```
WHILE <condition> DO
.
.
```

will be executed only if the condition is not satisfied when the loop is entered. See pages 22 and 23 of the Pascal User Manual and Report.

[05] Non-Apple Interface Card

Non-Apple interface cards are not recognized by the PASCAL system, since the correct PROM codes do not appear in the BIOS table. Enclosed is a copy of the listing of BIOS, which includes the driver routines used by PASCAL. Please understand that any modifications to BIOS are at the user's own risk.

(Please see the included App. Note #G10)

[06] System configuration to run Pascal

A 48K Apple II or Apple II Plus with at least one disk drive is required to run the Apple PASCAL system. Other Apple interface cards may be added for additonal system functions such as printer, external terminal, or modem.

[07] Terminal setup

(Please see included App. Note #G14)

[08] Intrinsic Units

(Please see included App. Note #G16)

[09] Include file

The PASCAL compiler "Include" directive (*$I<filename>*) causes the named file to be inserted into the compilation at that point. Similarly, the assembler directive ".INCLUDE <filename>" is used to insert files during assembly.

[10] Compile Swapping mode ($S+)

The PASCAL compiler swapping option (*$S+*) should be used if the compile fails and trashes the system. This option should be the first line of the text, preceding the program statement. Swapping MUST be used in compiling units. Please see page 91 of the Apple Pascal reference manual.

[11] Copy BASICS disk

The language system diskette named BASICS is a PASCAL-format diskette and must be copied using the PASCAL Filer Transfer.

[12] Extra linefeeds on printer

The double spacing effect on some printers when used with PASCAL can be remedied by e(X)ecuting APPLE3:LINEFEED. This program can be transfered to the diskette you boot on (usually APPLE1: or APPLE3:) as SYSTEM.STARTUP and it will execute automatically when the system is booted. I am also including the listing of LINEFEED for inclusion in your own programs.

(Please see the included App. Note #G2)

[13] Segment Procedures

(Please see included App. Note #G15)

[14] Compile with SYSTEM.LIST no code made ($L+)

The PASCAL compiler list option (*$L+*) should not be used, since this will result in the loss of the code file and possibly of the directory. Instead, direct the list to the printer (*$L Printer:*), the console (*$L Console:*), or to a named file ON ANOTHER VOLUME (*$L Mydisk:Myfile.text*). Use of the printer is recommended. This is a known error and will be corrected in the next release of PASCAL.

[15] Amount of free run-time memory

Approximately 39K bytes are available for program execution in the Apple PASCAL system, including the system and data segments required to run the program.

[16] Debugger

The "Debug" feature in PASCAL is not being supported by UCSD or Softech, Inc. and has not been implemented in the current version of Apple PASCAL.

[17] Save HIRES screen to disk

(Please see the included App. Note #G12)

[18] Filer transfer to PRINTER:

To transfer a text file to the printer, use the Filer "T" function. For example, to print the contents of GRAFDEMO.TEXT (on APPLE3), type "T", followed by "APPLE3:GRAFDEMO.TEXT,PRINTER:". The directory of APPLE3 can be printed out by typing "E", followed by "APPLE3:,PRINTER:".

[19] Micro-Modem support

Please contact Peripherals Unlimited for a utility that modifies the PASCAL operation system to allow the use of the Micro-Modem.

Peripherals Unlimited, Inc.
2633 East 28th Street, Suite 622
Signal Hill, CA 90806
(213) 595-6858

[20] Long Integer fix

An error in the implementation of PASCAL long integers results in a stack crash during a compare operation. The enclosed program will repair the LONGINTEGER module in SYSTEM.LIBRARY. This program should be run on all copies of the library, then saved for possible future use.

(Please see the included App. Note #G11)

[21] Real number format

Real numbers in PASCAL are formatted in four bytes (two words) in the following manner:

BIT	31	30....23	22.....0
ITEM	sign	exponent	mantissa

Double precision floating point numbers are not implemented in the current version of Apple PASCAL.

[22] Pascal exponents

The function 10 ** X is not part of the UCSD PASCAL definition ('10 ** X' means '10 raised to the X power'). The system intrinsic "PWROFTEN" returns 10 ** X, provided X is an integer in the range 0..37. Refer to page 138 of the white reference manual.

The function "EXP" (in TRANSCEN) is of the form e ** X, where X is a real number. The relationship between 10 ** X and e ** X is:

10 X = e (X ln 10) (ln = natural log)

[23] TTLOUT problem

The PASCAL TTLOUT procedure in the Applestuff unit of SYSTEM.LIBRARY does not function properly. The routine can be assembled and linked externally, using the listing and example shown on pages 100-106 of the Apple PASCAL Reference Manual. This is a known problem that will be corrected in the next release of PASCAL.

[24] No CHAIN in Pascal

PASCAL does not allow one program to call (run) another program. A program containing segment procedures can be used to simulate chaining.

[25] Capital letters required

UCSD PASCAL definitions require all compile-time options to be specified in capitals. This is not considered to be a bug.

[26] NOT problems

The boolean operator, NOT, does not invert the boolean variable correctly. For example, NOT FALSE doesn't equal TRUE. It is suggested that the 'NOT' be replaced by:

```
IF A=TRUE THEN A:=FALSE ELSE A:=TRUE
```

when working with booleans.

[27] SYSTEM.CHARSET

SYSTEM.CHARSET is a file of 1024 bytes, arranged in a logical sequence of 128 X 8 bytes, to represent the ASCII character set. The character is drawn from bottom to top, so byte 0 would be the bottom line of dots in the character. Drawblock uses these characters, so they should be accessed and treated as any other drawblock unit.

[28] CLEARSCREEN

The routine CLEARSCREEN as described in Bowles' text is a system function of the UCSD computer facility. On the Apple screen, this function can be emulated by using the command PAGE(OUTPUT), which issues a forms feed.

The following procedure will clear the screen on tha Apple video or an external terminal.

```
Procedure Clearscreen;
Begin
  Writeln(CHR(12));
End;
```

NOTE: CHR(12) is the character used to clear the video screen. The character used should be the "erase screen" parameter as shown in SETUP.

[29] CALC

APPLE3:CALC.CODE allows the evaluation of simple arithmetic expressions, using +,-,*,/ and (). Trignometic functions and exponentiation will cause the program to abort with an "unimplemented instruction" error at this time. To exit the program, simply press "RETURN" in response to the prompt.

For example:

```
->45*(-2)
    -9.00000E1
```

DOS

C. DOS

[01] How to BLOAD/BSAVE

The use of the disk to store, load and run machine language programs is explained in chapter 9 of the DOS 3.2 manual (pages 92 and 93).

[02] Printer-DOS disconnect

Because DOS 3.2 uses the normal Apple I/O to intercept commands from BASIC, the PR# and IN# statements take on new significance. The usual problem is that the DOS doesn't work after you turn the printer on or off. Please read pages 100 to 102 of the DOS 3.2 manual for a complete discussion of these effects and how to avoid them.

[03] Crashed disk recovery

There are three basic ways for the information on your diskette to become damaged:

1) Destruction of data - through damage to media or accidential erasure.
2) Damage to the Directory information or the Volume Table of Contents (VTOC).
3) Damage to the DOS itself.

Lets take these in reverse order...

If the DOS is damaged you will usually find that the diskette will not boot but that programs and files can be accessed by booting another (undamaged) diskette, inserting the first one and proceding normally from there. You can sometimes recover from this problem by running the UPDATE 3.2.1 program on the offending diskette. If this doesen't work you will have to transfer the files one-by-one to a newly initialized diskette.

Damage to the directory or VTOC is a little more complex but still (usually) recoverable. To do this you must obtain one of the disk utility programs currently on the market and physically repair the damage. You will find Appendix C of the DOS 3.2 Reference Manual especially helpful.

If the data itself is damaged or the diskette partially erased you will, in all likelyhood, not be able to recover. Make frequent backup copies of your work to guard against this possibility.

[04] Software slot finder

DOS 3.2 and 3.2.1 both keep a table of the current system parameters (DOS 3.3 does too). Here are the addresses of the most useful for a 48K Apple. Subtract 16384 for a 32K Apple.

```
CURRENT VOLUME         = PEEK (43622)
CURRENT DRIVE          = PEEK (43624)
CURRENT SLOT           = PEEK (43626)
CURRENT RECORD LENGTH  = PEEK (43628)
CURRENT RECORD NUMBER  = PEEK (43630)
CURRENT BYTE IN RECORD = PEEK (43632)
```

[05] 6502 interfacing

There are two ways to use DOS from machine language, Read Write Track Sector and the normal commands. Please refer to pages 94 to 98 of the DOS 3.2 manual for more information on RWTS. The other way is to send the normal DOS commands, one character at a time in the 6502 accumulator, to $FDED. Remember to precede the commands with a carriage return and control-D and follow it with another carriage return. Here is a short-cut if you have ROM Applesoft.

```
300:LDA #$20
302:LDY #$03
304:JSR $DB3A ;Applesoft's string printing routine
307:RTS

320:0D 04 43 41 54 41 4C 4F 47 0D 00
          C  A  T  A  L  O  G
```

If you use the indirect only commands you must also convince DOS that BASIC is running.

From Applesoft $76 must not contain $FF
and $33 must not contain $DD
From Integer $D9 must be greater than $7F

[06] BLOAD A,L parameters

Immediately after BLOADing a program you can find the A and L parameters with which it was BSAVEd by using the pointers explained on page 144 of the DOS 3.2 manual.

[07] Direct vs Indirect commands

DOS 3.2 and 3.2.1 check to see if the Applesoft program is running before it will execute certain commands. If you try to restart an Applesoft program with CONT or GOTO, one of the "running" flags may not be set. So instead type POKE51,0:CONT, POKE51,0:GOTO 100, or have the POKE51,0 in the program before the next DOS command.

[08] APPEND overwrites start of file

Enclosed is information concerning APPEND in DOS 3.2 and 3.2.1.

(Please see included App. Note #J2)

[09] Reset re-boots the disk

Booting with DOS 3.1 in a system with an Auto-Start ROM will cause a re-boot whenever reset is pressed. To prevent this you can either update the diskette to DOS 3.2.1 (see the DOS manual) or include the following in your HELLO program:

```
POKE 1010,208
POKE 1011,3
POKE 1012,166
```

[10] DOS commands and Applesoft GET

The control-D of a DOS command must be the first character on a output line. This means that it must be immediately preceeded by a carriage return. If it isn't, the command will be printed out to the screen or the printer and ignored by DOS. The most common cause for this is having a GET statement or a PRINT statement that ends with a semi-colon or comma. Another PRINT with no semi-colon or comma will generate the carriage return required for the next DOS command to be recognized.

Mini-Assembler

D. MINI-ASSEMBLER

[01] How to use

The Mini-Assembler is a convenient development tool for small programs. For major programming jobs it is suggested that you get a full sized assembler. To use the Mini-Assembler you must first be in Integer BASIC. If you have an Apple II Plus then you will need a firmware card or a Language card. Enter the monitor with CALL -151 and type F666G. The included document will take you from there.

(Please see the included App. Note #C3)

Apple II Plus

E. APPLE II PLUS

[01] Mini Assembler not there

The Apple II mini-assembler is not found in the Apple II Plus because it was written to fit in a hole in the Integer BASIC ROMs. I am including the RAM version of the mini-assembler for your use.

(Please see included App. Note #C7)

[02] How to run an Integer Basic program

There are no cassette or disk versions of the INTEGER BASIC Interpeter currently available from Apple Computer. To run an INTEGER BASIC program on an Apple II Plus you will need to do one of three things.

1) Install an INTEGER BASIC ROM Card in slot zero.
2) Install a Language Card in slot zero (product number A2B0006; 48K RAM and disk required).
3) Convert the INTEGER BASIC program to an Applesoft Program. (See Note #B9)

[03] APPLE II vs APPLE II Plus

The only difference between the Apple II and the Apple II Plus is that the Apple II has Integer BASIC and the "old" monitor ROM while the Apple II Plus has Applesoft BASIC and the Autostart monitor ROM. Most of the game programs available today are written in Integer BASIC and most of the business, scientific, and indusrtial programs require Applesoft, so the selection depends on your application. Apple offers firmware cards that will supply whichever BASIC your machine lacks.

Monitor ROM

F. MONITOR ROM

[01] S and T commands in Autostart ROM

To make room for the autostart and enhanced editing features in the new monitor ROM Single Step and Trace were removed. (Please see the new Reference Manual for listings and details.)

[02] Accessing the monitor

To enter the monitor from either BASIC type:

```
CALL -155 <RETURN>
```

To get the Apple to go into monitor mode upon RESET, do a CALL -155 to enter the monitor. After the monitor prompt (*), type:

```
3F2:69 FF 5A <RETURN>
```

and press RETURN. Then the Apple will go into monitor mode whenever you press RESET.

[03] Autostart ROM reset vector

The Autostart ROM allows the machine language programmer to execute his own routine after a RESET instead of automatically dropping into the monitor or BASIC. The (new) Apple II Reference Manual describes how to use this feature on pages 36 and 37.

Firmware Card

G. FIRMWARE CARD

[01] What the switch does

The switch on the rear of the firmware card determines which set of ROMs will be used immediately after a reset. If the switch is up then the ROMs on the card will be used, otherwise the ROMs on the main board will be used. DOS confuses things by forcing the version of BASIC it was in last every time it gets control, regardless of the position of the switch and DOS gets control immediately after a reset in a system with an Auto-Start ROM.

[02] Contact oxidation

Occasionally, peripheral cards in the Apple collect some oxidation on the contact fingers which can cause an intermittent connection. This can result in various system errors.

To clean off contacts, turn off the power and remove the cards. Using a soft pencil eraser ("Pink Pearl" or such), gently clean off the contacts. Replace the boards, seat firmly, then reboot the system. If this does not correct the problem, contact your dealer for assistance.

Language Card

H. LANGUAGE CARD

[01] Do you get both Basics with language card?

The Language System inculdes a diskette marked "BASICS". When you boot with this diskette a program is run that looks to see which BASIC, Applesoft or Integer, is in the machine and then loads the Language card with the other language. This means that whichever BASIC you have, you will get the other. In addition, if you have an Apple II Plus, the code for the Programmer's Aid #1 is also loaded into the card.

[02] Microchess doesn't work

We are aware that Microchess will not work with the Language system. The problem is that the P5A PROM on the disk controller card forces you to use the BASICS diskettte and the BASICS diskette skips part of the normal DOS boot code which Microchess uses for its software protection scheme. When we have a fix we will let you know through your dealer and CONTACT.

[03] Programmers Aid #1?

When you boot with the BASICS diskette on your Apple II Plus, you automatically load the Programmers Aid programs into the Language card with the Integer BASIC. See your dealer for the Programmer's Aid manual. Regular (non-Plus) Apples will still need to plug the ROM into socket D0.

Cassette Interface

I. CASSETTE INTERFACE

[01] Cassette routines

Included is a description of the cassette tape routines in the Apple monitor and the Auto-Start ROM.

(Please see the included App. Note #D2)

[02] Tape loading problems

Tape load problems might be caused by several things:

1. Bad tapes
2. Misaligned recorder or player
3. Magnetic interference from a TV being too near the recorder
4. Electrical noise on the power line generated by household light dimmers
5. A possible ground loop within the recorder (try unplugging the microphone jack while loading)

Printer Interfacing

J. PRINTER INTERFACING

[01] Controlling a printer from within a Basic program

The Apple has one output channel that can be directed to the screen or any of the peripheral slots (except slot 0). To change where the output goes use PR#n (where n is the slot number and 0 means the screen). Please note that PR# is also a DOS command and if you are using disk, it must be used in a PRINT with a control-D. Example:

```
10 PRINT "TO THE TV."
20 PRINT CHR$(4);"PR#1"
30 PRINT "TO THE PRINTER IN SLOT 1."
40 PRINT CHR$(4);"PR#0"
50 PRINT "BACK TO THE TV."
60 END
```

Please read pages 100-102 of the DOS 3.2 manual for more information about PR# and DOS.

[02] Centronics 730 normal/expanded printing

The Centronics 730 uses bit 8 to control the expanded print option but the Apple normally always sends that bit set, which means that the printer is always in expanded mode. I am including a program that allows software control of bit 8.

(Please see the included App. Note #E5)

[03] Tabbing on printer

Apple's interface cards are programmed to make nice listings of BASIC programs. Unfortunately this feature interfers with the TAB function when used with the printer. So instead of:

```
100 PRINT "HELLO"; TAB(70); "THERE!" use:

100 PRINT "HELLO";: POKE 36,70 : PRINT "THERE!"
```

You can follow the last quote with another ";:" to continue tabbing.

[04] 6502 printing

To print characters to the Apple screen or other peripheral devices simply requires sendint the ASCII with the most significant bit set one character at a time in the 6502 accumulator to $FD18. Here is a short-cut if you have ROM Applesoft.

```
300:LDA #$20
302:LDY #$03
304:JSR $DB3A
307:RTS

320:48 45 4C 4C 4F 00
    H  E  L  L  O
```

To change the slot you can load the accumulator with the slot number and JSR to $FD95 unless you are using DOS. With DOS you must print a carriage return, control-D, "PR#", the slot number in ASCII and another carriage return.

[05] Centronics 779 gives extra linefeeds

There are two configurations of Centronics 779 printer and two configurations of Apple parallel printer interface.

	Centronics version	General Purpose version
PROM number	P9-00	P1-02
779 with auto-linefeed	normal	double spacing
779 without auto-linefeed	over-printing	normal

Inside the Centronics 779 between ME13 and ME14 there is a jumper area marked E1, E2, and E3. See page 3-10 of the 779 techinical manual for instruction on changing the auto linefeed option.

(Please refer to Applesource 8, page 3)

[06] Centronics 779 gives no linefeeds

(Please see the above note)

Serial Card

K. SERIAL CARD

[01] Printer handshaking

The Apple High Speed Serial Interface card does not have built in handshaking. This causes many printers to loose characters when running faster than 300 baud. However, Diablo, Qume, or NEC Spinwriters can get P8A PROM which will implement a software handshake with these printers. I am enclosing information about using the data input line for handshaking with other printers.

(Please see included App. Note #E3)

[02] Line width > 40 trashes program (Applesoft)

If you are using a printer with the line width set to greater than 40 and also sending the information to the Apple screen, you may wipe out your Applesoft program. Apple's screen is memory mapped and an offset corresponding to a cursor position greater than 40 will cause the information to be "printed" in the same memory as the start of your Applesoft program. Therefore it is suggested that you use the printer or the video but not both at the same time.

Communications Card

L. COMMUNICATIONS CARD

[01] Download from mainframe to text file

When downloading programs or data from a remote computer to an Apple II you will lose data whenever the Apple is forced to ignore the com card - either to process the incoming data or write it to an I/O device (Disk, Printer).

The solution to this problem is to buffer the incoming data. You can do this by:

1) Requesting fixed size blocks of data from the remote machine - Reading in all of one block and processing it before requesting the next one.

2) Controlling (if possible) the data flow from the remote machine with XON (control S) and XOFF (control Q). recieving data until the buffer is almost full, transmitting XOFF to halt the data flow, processing the data, and transmitting XON to resume the process.

[02] SOROC & Applesoft, no linefeeds

When attempting to use a SOROC IQ120 as a remote terminal from Applesoft all of the output will be printed on one line of the SOROC screen. The solution to this problem is to use the Com Card Print routine on page 28 of the Communications Interface Card manual. If you are using DOS 3.2 or later you must modify this routine with a POKE 845,110 BEFORE executing the CALL.

[03] Modify ACIA status

Modifying the default parameter settings on the Communications card is described on page 27 of the Communications Interface Manual. But please note that the PR#n and IN#n commands do not initialize the interface, they only reset the I/O vectors. It's the first character through the interface that loads the default parameters. So be sure to PRINT or INPUT at least one character before doing the POKEs.

[04] Datamover problem

The Datamover program supplied with the Communications Interface card was not designed to work with the Applesoft language. Several packages are available commercially that will allow the transfer of data.

Two of these companies are:

Peripherals Unlimited
3450 E. Spring Street Suite 206
Long Beach, CA 90806
(213) 595-6858

Southeastern Software
7270 Culpepper Drive
New Orleans, LA 70126
(504) 246-8438
(504) 246-7937

[05] 4800 baud missing wire

The Communications Card is wired to operate at 110 and 300 baud. In the back of the Comm Card Manual there are instructions for modifying the Comm Card for higher baud rates. The drawing on page 36 for 4800/1200 baud is missing a wire. In addition to the wires in the drawing, connect pin 15 on the socket of A1 to pin 15 of the socket of A2.

[06] Print routine fix for 3.2 or 3.2.1 DOS

The Communications Card does not generate a linefeed with a carriage return. On page 28 of the Comm Card Manual there is a "Printer Driver" which will supply linefeeds with carriage returns. Unfortunately the driver was written for DOS 3.1. To make the driver work with DOS 3.2 you need to enter the driver into memory and type:

```
34D:6E   from the monitor or
POKE 845,110  from BASIC
```

then save and use the driver according to the directions on page 29.

[07] Communications Card handshake

If you have a printer that requires handshaking you can have your Apple Communications Card modified by our service department to accept the proper signals. There is a service charge of $35. Please contact the Apple Service Department for further details.

[08] <CTRL-R> for remote terminal

To use your Apple and communications card from a remote terminal use the following steps:

On the Apple send: PR# <slot>
IN# <slot>

<Slot> is the number of the Apple I/O slot in which the Comm Card is inserted.

Or, if the Apple is acting as a "dumb terminal", from the external terminal send a CTRL R and a PR# <slot>.

To reverse this procedure:

From the terminal send: PR#0
IN#0

Or, to return the Apple to "dumb terminal" mode, send a CTRL T from the remote terminal.

[09] How to use with a modem

Using the Comm Card with a modem is usually quite simple. Here is a quick example of how it's done...

Type: IN# <slot> using the slot number in which the Comm card is connected.

Then: <CTRL> A
<CTRL> F "Full Duplex" remote machine echos characters

Or: <CTRL> A
<CTRL> H "Half Duplex" Apple echos characters

The Com Card normally operates at 30 characters per second (300 Baud). If you are connecting to a machine that runs at only 10 characters per second (110 Baud) type:

<CTRL> A
<CTRL> 1 Set 10 CPS

Now turn on the modem, dial the remote computer and place the phone handset in the modem. You are now connected to the remote machine.

To send a BREAK character type:

```
<CTRL> A
<CTRL> S  Start BREAK signal
```

Type any other character to stop sending BREAK.

To disconnect your Apple from the other computer, type:

```
<CTRL> A
<CTRL> X
PR#0
```

and hang up the phone.

[10] 9600 baud modifications

If you need to communicate with a remote device at 9600 BAUD and happen to have an extra Com Card you can make the attached modifications. NOTE: This modification invalidates your warranty.

(Please see included App. Note #A5)

Parallel Printer Card

M. PARALLEL PRINTER CARD

[01] Configuration block

Throughout the Parallel Interface Manual, a negative going acknowledge is shown as ACK (with a bar over it) and a positive going acknowledge as ACK (without the bar). Therefore, when choosing a jumper block for printers that have a negative going strobe and negative going acknowledge signal find the jumper block on page 6 that is labeled (bar STR)-(ACK).

General Interfacing

N. GENERAL INTERFACING

[01] 20 ma current loop

The only interface that Apple markets with a 20ma. current loop is our High Speed Serial Interface. It has an active send loop and a passive receive loop. We can supply instructions on converting the receive loop to active. Please see our catalog or your dealer for more information.

[02] Interface to TV station

The Apple II produces NTSC compatible video. However it isn't NTSC standard video. The only way we know of to broadcast the Apple's video is to aim a camera at the a video monitor. We don't know of anyone at this time who has sucessfully used a Time Base Corrector or modified the Apple to conform to NTSC.

[03] TTY driver

The TTY driver routine in the old Apple II Reference manual doesn't work with DOS in the system. The enclosed document tells how to update the software driver to work with DOS. Be sure to see the note at the end or the text explaining how to make the driver work with DOS 3.2 or DOS 3.2.1.

(Please see the enclosed App. Note #E4)

[04] Card reader sources

Optical mark readers are available from the following companies:

Chatsworth Data Corp.
20710 Lassen Street
Chatsworth, CA 91311
(213) 341-9200

Scantron Corp.
PO Box 45706
8820 South Sepulveda Blvd.
Los Angeles, CA 90045

[05] Numeric keypad

An Apple-compatible numeric keypad is currently available from:

Advanced Business Technology, Inc.
12333 Saratoga-Sunnyvale Rd.
Saratoga, CA 95070
(408) 446-2013

This unit connects to the existing Apple keyboard and is provided with a 6-foot cable.

Hardware

0. HARDWARE

[01] Eurapple conversion

Conversion of an American (NTSC) Apple to European (PAL or SECAM) television standards is not recommended. There are several circuit modifications invloved. It is much better to buy the type that you will need in the first place. An American Apple can be used in Europe with suitable voltage correction equipment and an NTSC television. The Apple will work as well with 50Hz as 60Hz.

[02] Apple power consumption

The Apple power supply's power consumption is rated at 60 Watts at full load. Please see page 92 of the new Apple II Reference manual for more details.

[03] Non-Apple memory expansion-speed

The Apple II and Apple II Plus use 4116 compatible memory with an access time of less than 250 nanoseconds. See page 12 of the October-November 1978 issue of MICRO for a list of manufacturers part numbers and access times.

[04] Interrupts

The Apple II does not use interrupts for any of it's normal operations. There are two points to remember when you use interrupts. First, use IRQ and not NMI if there will be a disk in the system. Second, when the 6502 vectors through $3FE and $3FF, the accumulator has already been saved at $45 and the present contents are trash, so restore the accumulator from $45 before the RTI. If you are careful about restoring all the registers, the interrupts will work just fine.

Video

P. VIDEO

[01] 80 chars per line on Apple screen

The Apple II video screen is hardware-defined as 24 lines of 40 characters per line. The following companies are advertising a plug-in board for 24 lines of 80 characters for the Apple:

M & R Enterprises
PO Box 61011
Sunnyvale, CA 94088
(408) 738-3772

The Computer Stop
16919 Hawthorne Blvd.
Lawndale, CA 90260
(213) 371-4010

VIDEX
3036 N.W. Thistle Place
Corvallis, OR 97330
(503) 758-0521

Please contact these firms for more information.

Hires

Q. HIRES

[01] Character sets

The Apple cannot display normal text on the graphics screen but characters can be drawn on the HIRES screen. The HI-RES CHARACTER GENERATOR is described in the Apple Software Bank manual entitled "Contributed Programs Volumes 3-5" (product# A2L0014) on pages 20-28. Both are available at your dealer.

[02] Shape table BSAVE & BLOAD

Please refer to the enclosed for information about using BSAVE for Applesoft shape tables.

(Please see included App. Note #B7)

[03] Screen mapping & colors

The Apple HIRES screen cannot display all colors at all locations. This causes colored vertical or near vertical lines to appear as dashed lines or not appear at all. This effect is explained more throughly in the (new) Apple II Reference Manual on page 19.

[04] HGR trashes RAM Applesoft

RAM Applesoft (on disk or cassette) is loaded into decimal memory locations 2048 through 12288. Since the high resolution graphics screen begins at decimal address 8192 and ends at decimal address 16383, there is a memory conflict between RAM Applesoft and Hi-Res graphics page one. Therefore, executing an HGR statement will clear the Hi-Res page and also part of the Applesoft interptreter itself. The result will disasterous and you will have to reload the interpreter and, if you have a disk, re-boot the system. The only ways around this are to always use page two of Hi-Res (HGR2) or to install an Applesoft ROM Card your system.

[05] Plotting on both screens

Plotting on both HIRES screens from Applesoft is relatively simple. The POKEs listed on pages 132 and 133 of the Applesoft Reference Manual control which screen is displayed and location 230 (decimal) controls which screen will be modified by the HIRES commands. POKE 230,32 will cause Applesoft to draw on screen 1 (HGR) and POKE 230,64 will draw to screen 2 (HGR2).

Apple Post

R. APPLEPOST

[01] Non-Apple printer card

Apple Post examines the machine code on the printer interface card to determine what type of card it is (serial, parallel). If you have a non-Apple printer card, Apple Post won't recognize it and will not allow label printing. If you have this problem, the following may be of some help.

```
Parallel = card type 1
Serial   = card type 2
Comm     = card type 3
```

In the Utility Module:

```
13135 V= <card type>
```

In the Output Module:

```
1125 X2= <card type>
```

Be sure to make backup copies of your diskettes before making ANY changes to the programs!

[02] Used the sample program

The Sample data file in Apple Post is for example purposes only. It has room for only a few names and should not be used with the ENTER command (as stated on page 5 of the manual).

If you have entered data into this file (and you couldn't have entered much), you'll have to start over with a correct data diskette.

Renumber

S. RENUMBER

[01] Renumbers after an asterisk "*"

Renumber is a very powerful tool for developing programs but if you use it and you find some strange alterations in you program, Renumber may have done it. What happens is that the number after a "*" sometimes is mistaken as a line number and Renumber renumbers it. So if you had a line:

```
10 LET A = B * 10
```

it might renumber as:

```
20 LET A = B * 20
```

The fix is:

For RAM Applesoft	For ROM Applesoft
]LOAD RENUMBER	]LOAD RENUMBER
]POKE 14342,172	]POKE 4815,172
]POKE 14343,171	]POKE 4816,171
]SAVE RENUMBER	]SAVE RENUMBER

Miscellaneous

T. MISCELLANEOUS

[01] RAM test of top 16K with DOS

Testing the RAM in the top row of memory in a Apple with a disk requires that the DOS be disabled, otherwise the Apple hangs and reset is the only way to regain control. To disable DOS, get into the monitor (CALL -151) and type FE89G <RETURN>, FE93G <RETURN>. You will have to re-boot the disk when you are done.

[02] Back issues of CONTACT

Back issues of CONTACT are not available. However, BEST OF CONTACT '78 (Part No. A2L0020) contains volumes 1-4, and is available at your local dealer.

[03] PEEKs, POKEs, and CALLs

PEEK, POKE, and CALL all refer to machine language programs or data being used from a BASIC program. Often their use cannot be explained because a machine language program is loaded with or generated from the specific BASIC program. There is a list of general purpose PEEKs, POKEs, and CALLs in the (new) Applesoft II Reference Manual starting on page 128. The same information can be found in less convenient forms in the (new) Apple II Reference Manual.

[04] Availability of source code and schematics

Most of the source code and schematics for Apple Computer's products are included in the manual sent with the product. However the following items are proprietary and won't be sent out.

1) Source code for PASCAL
2) Source code for Applesoft
3) Source code for DOS 3.2
4) Schematic for the Language card
5) Schematic for the Firmware card
6) Service diagnostic package

Application Notes

```
]LIST

10  REM **********************
20  REM *                    *
30  REM *  SIMPLE TONES FOR  *
40  REM *                    *
50  REM *    APPLESOFT II    *
60  REM *                    *
70  REM *     J CROSSLEY     *
80  REM *                    *
90  REM **********************
100  REM
110  REM   THIS IS THE ROUTINE
120  REM FROM PAGE 45 OF THE
130  REM 'RED BOOK' MODIFIED
140  REM FOR APPLESOFT.
150  REM
160  HOME
170  LIST  - 150
180  REM
190  REM    INSERT THE FOLLOWING
200  REM ROUTINE IN YOUR PROGRAM
210  REM THEN POKE 768,TONE
220  REM      POKE 769,DURATION
230  REM AND, CALL 770 FOR TONES
240  REM
250  FOR I = 770 TO 790
260 :: READ J
270 :: POKE I,J
280  NEXT
290  DATA 173,48,192,136,208,5
300  DATA 206,1,3,240,9,202,208
310  DATA 245,174,0,3,76,2,3,96
320  FOR TN = 100 TO 10 STEP  - 1
330 :: POKE 768,TN: REM TONE
340 :: POKE 769,10: REM DURATION
350 :: CALL 770
360  NEXT
370  HOME
380  LIST 180,360

]
```

USING APPLESOFT SHAPE TABLES FROM DISK

SHAPE LOADER

The Applesoft SHLOAD statement loads shape tables from cassette tape and sets up the proper pointer so that Applesoft can use it. The following program loads the shape table from the disk as a binary file and sets up the proper pointer for Applesoft. The INPUT (line 100) and BLOAD (line 110 and 140) statements assume that the binary filename is prefixed by 'SHP.' because that is the format the SHAPE SAVE program on the next page uses. This routine should be the first thing in the program since it destroys the string variables in the process of loading the shape table.

```
100  INPUT "WHICH SHAPE TABLE TO LO
     AD ?  SHP.";F$
110  PRINT  CHR$ (4);"BLOAD SHP.";F
     $;",A$D000"
120 DS =  PEEK (977) +  PEEK (978) *
     256 + 3233
130 HM =  PEEK (115) +  PEEK (116) *
     256 -  PEEK (DS) -  PEEK (DS +
     1) * 256
140  PRINT  CHR$ (4);"BLOAD SHP.";F
     $;",A";HM
150  HIMEM: HM
160  POKE 232,HM -  INT (HM / 256) *
     256: POKE 233,HM / 256
```

SHAPE SAVER

This program saves your shape table on the disk as a binary file for the SHAPE LOADER routine to use. You should load this program before getting into the monitor and typing in your shape table. The best place to start your shape table is at $2000. That leaves lots of room for the shape saver and the table. Use the instructions in chapter 9 of the Applesoft Reference manual. When you've got it all typed in and checked, type 3D0G to get back into Applesoft and RUN the program.

```
1000  REM  SHAPE SAVER
1005 H$ = "0123456789ABCDEF"
1010  TEXT : HOME : VTAB 5
1020  INPUT "WHAT IS THE STARTING A
     DDRESS (HEX)?";A$
1030  IF  LEN (A$) = 0 THEN  END
1040  GOSUB 5000
1050  IF F < 0 THEN 1000
1060 S = X
1070  PRINT
1080  INPUT "WHAT IS THE ENDING ADD
     RESS (HEX)?";A$
1090  IF  LEN (A$) = 0 THEN 1000
1100  GOSUB 5000
1110  IF F < 0 THEN 1000
1120  LET E = X
1130  IF E <  = S THEN  PRINT  CHR$
     (7);"THE END IS LESS THAN THE
     START!": GOSUB 5100: GOTO 1000

1140  PRINT
1150  PRINT "WHAT NAME DO YOU WANT
     TO USE ?"
1160  INPUT "     SHP.";A$
1170  PRINT  CHR$ (4);"BSAVE SHP.";
     A$;",A";S;",L";E - S + 1
1180  END
5000  PRINT : PRINT "CONVERTING TO
     DECIMAL"
5010  LET F = 0:X = F
5020  FOR J = 1 TO  LEN (A$)
5030  FOR I = 1 TO 16
5040  IF  MID$ (A$,J,1) =  MID$ (H$
     ,I,1) THEN X = X * 16 + I - 1:
     F = F + 1
5050  NEXT I,J
5060  IF F <  >  LEN (A$) THEN F =
      - 1:"THAT'S NOT HEX!!!"; CHR$
     (7): GOSUB 5100
5070  IF X > 2 ^ 16 THEN F =  - 1: PRINT
     "THAT'S TOO BIG!!!"; CHR$ (7):
      GOSUB 5100
5080  RETURN
5100  FOR J = 1 TO 1000: NEXT J: RETURN
```

```
 LIST

100  REM          PRINT USING
110  REM
120  REM           SIMULATOR
130  REM
140  REM             AUG 79
150  REM
160  REM          J. CROSSLEY
170  REM
180  LET N = 2: REM SET NUMBER
190  REM             OF DECIMALS
200  LET S = 5: REM SET FIELD
210  REM             WIDTH
220  HOME
230  FOR X =  - 5 TO 5 STEP .501
240  PRINT X,"$";
250  GOSUB 2000
260  PRINT
270  NEXT X
280  PRINT
290  PRINT "UNFORMATTED     FORMATTED"
300  END
1000  REM THIS IS THE FORMATTING
1010  REM SUBROUTINE. THE INPUT
1020  REM IS 'X','N', AND 'S'
1030  REM X IS THE NUMBER TO BE
1040  REM   BE PRINTED
1050  REM N IS THE NUMBER OF
1060  REM   DIGITS RIGHT OF '.'
1070  REM S IS THE WIDTH OF THE
1080  REM   RIGHT JUSTIFIED
1090  REM   PRINTING FIELD
1100  REM
2000 X$ = " " +  STR$ ( INT (X * 10 ^ N + .5))
2010 Q =  LEN (X$) - ( VAL (X$) < 0)
2020  PRINT  SPC( S - Q * (Q > N + 1) - (N + 2) * (Q <  = N + 1));
2030  PRINT  MID$ (X$,1 + ( VAL (X$) < 0),(Q <  = N) + (Q - N) * (Q > N))
     ;
2040  PRINT  MID$ ("0.00",1 + ((N + 1) < Q),1 + (N - Q + 2) * (Q < N + 2)
     );
2050  PRINT  RIGHT$ (X$,N * (Q > N) + (Q - 1) * (Q <  = N));
2060  RETURN

]
```

CONVERTING INTEGER BASIC PROGRAMS TO APPLESOFT

Page 76 of the DOS 3.2 Reference Manual contains a routine which can be used to convert Integer Basic programs to Applesoft using the disk system. Integer Basic is required for the first step of the conversion only.

First, load in the Integer program. Add the routine on page 76 at the very beginning of the program, adjust the LIST line for the specific program, and give the textfile a unique name. Then type "RUN". This will cause the program to be listed out to the disk as a textfile under the name specified. At this point, the text file is independent of the language.

To convert, type "EXEC <your filename>" in Applesoft. The textfile will be read into the system as if typed from the keyboard. Since Applesoft does not check syntax at input time, the Integer syntax is accepted, although the program may not run. Save the program in Applesoft as usual.

Micro Magazine, January 1980, contains an article describing Integer to Applesoft conversion using cassette instead of disk. The article also describes some of the syntax and logic changes that will usually have to be made.

APPLESOFT RANDOM NUMBERS

The Applesoft random number generator, like all such routines, is only a pseudo-random generator, so non-random patterns will eventually occur. The frequency of repetition of these patterns will vary from procedure to procedure. Proper re-seeding of the random number generator during a program will help prevent the appearance of large repeating sequences. This can be done in two ways, and for best results, both should be used.

1. Seed the random number calculation at the beginning of the program, using the keyboard count location. This will take the form

```
S = PEEK(78) + PEEK(79)*256
X = RND(-S)
```

2. Within the random generating portion of the program, insert a statement of the form Z = RND(-RND(9)), which will begin a new random sequence. (See page 102 of the Applesoft Reference Manual.)

Please be aware that no method will completely eliminate patterns in the random numbers generated, but we can break up the sequences so that objectional non-random patterns are less likely to appear.

OUT OF MEMORY ERRORS

There are two ways to get an 'OUT OF MEMORY ERROR' in Applesoft. The most obvious cause is to have a program that is too big or uses too many variables. The only solution in this case is to trim down the program, keep the data on a disk, chain the program in from the disk in segments, or upgrade your system to 48K.

The less obvious cause is stack overflow. This is easy to spot because after getting the OUT OF MEMORY error, PRINT FRE(0) tells you that there is still free memory and, since the error clears the stack, everything else seems normal. The problem is that Applesoft uses the 6502 stack to save it's recursive subroutine calls and the stack is a limited resource. Here are some causes:

* Too many levels of FOR-NEXT loop
* Too many levels of GOSUB
* Excessively complex mathematical or string formulas
* GOSUBs with no RETURN
* Improper recovery in ONERR GOTO routines
* CALLs or interrupts that don't restore the stack properly

NOTE: THESE EFFECTS ARE CUMULATIVE. You might be affected by more than just one.

The first four are inherent in your program structure. If your program is cleanly structured then these probably won't cause trouble.

If you are using ONERR GOTO you should carefully examine pages 81, 82, and 136 of the Applesoft Reference manual. There are two correct ways to leave an ONERR routine. You can use RESUME which takes care of the stack and re-executes the statement that caused the error or you can use the stack recovery routine on pages 82 and 136 (the example on page 136 is easier to use) before you do a GOTO to a line number. Beware, this recovery routine does not clear any GOSUBs or FOR...NEXT loops!

When Applesoft executes a CALL, it does a 6502 JSR to the specified address. It's up to you to pull off anything you pushed on the stack before the RTS. Likewise the routine to handle an interrupt from a peripheral card must restore the stack and the registers.

A very large, complex program may be forced into a stack overflow condition by the user's responses. CALL 54915 will clear the stack without hurting the variables, but it wipes out all pending FOR-NEXT loops, GOSUBs and formulas. This allows a program controlled restart. Beware, this is no substitute for good programming practices but a way to recover when a program exceeds the Apple's capibilities.

APPLE II MINI ASSEMBLER F666G

The following section covers use of the Apple II mini-assembler only. It is not a course in assembly language programming. For a reference on programming the 6502 microprocessor, refer to the MOS Technology Programming manual. The following section assumes the user has a working knowledge of 6502 programming and mnemonics.

The Apple II mini-assembler is a programming aid aimed at reducing the amount of time required to convert a handwritten program to object code. The mini-assembler is basically a look-up table for opcodes. With it, you can type mnemonics with their absolute addresses, and the assembler will convert it to the correct object code and store it in memory.

Typing "F666G" will put the user in mini-assembler mode. While in this mode, any line typed in will be interpreted as an assembly language instruction, assembled, and stored in binary form unless the first character on the command line is a "S".

If it is, the remainder of the line will be interpreted as a normal monitor command, executed, and control returned to assembler mode. To get out of the assembler mode, reset must be pushed.

If the first character on the line is blank, the assembled instruction will be stored starting at the address immediately following the previously assembled instruction. If the first character is nonblank (and not "S"), the line is assumed to contain an assembly language instruction preceded by the instruction address (a hex number followed by a ":"). In either case, the

instruction will be retyped over the line just entered in disassembler format to provide a visual check of what has been assembled. The counter that keeps track of where the next instruction will be stored is the pseudo PC (Program Counter) and it can be changed by many monitor commands (eg. 'L', 'T', . . .). Therefore, it is advisable to use the explicit instruction address mode after every monitor command and, of course, when the Tiny assembler is first entered.

Errors (unrecognized mnemonic, illegal format, etc.) are signalled by a "beep" and a carrot ("^") will be printed beneath the last character read from the input line by the mini assembler.

The mnemonics and formats accepted by the mini assembler are the same as those listed by the 6502 Programmers Manual, with the following exceptions and differences:

1. All imbedded blanks are ignored, except inside addresses.
2. All addresses typed in are assumed to be in hex (rather than decimal or symbolic). A preceding "S" (indicating hex rather than decimal or symbolic) is therefore optional, except that it should not precede the instruction address).
3. Instructions that operate on the accumulator have a blank operand field instead of "A".
4. When entering a branch instruction, following the branch mnemonic should be the target of the branch. If the destination address is not known at the time

the instruction is entered, simply enter an address that is in the neighborhood, and later re-enter the branch instruction with the correct target address. NOTE: If a branch target is specified that is out of range, the mini-assembler will flag the address as being in error.

5. The operand field of an instruction can only be followed by a comment field, which starts with a semi-colon (";"). Obviously, the Tiny assembler ignores the field and in fact will type over it when the line is typed over in disassembler format. This "feature" is included only to be compatible with future upgrades including input sources other than the keyboard.
6. Any page zero references will generate page zero instruction formats if such a mode exists. There is no way to force a page zero address to be two bytes, even if the address has leading zeroes.

In general, to specify an addressing type, simply enter it as it would be listed in the disassembly. For information on the disassembler, see the monitor section.

APPLE II PLUS MINI-ASSEMBLER

The attached listing is a relocated version of the mini-assembler for use with the Apple II Plus and the instructions for the Apple II version. The instructions are the same except how to enter the mini-assembler. This version can be BRUN fron the disk or BLOADed and start with CALL 2048 from Applesoft. To restart use CALL 2051. From machine language the start is 800G and the restart is 803G.

Please note that the mini-assembler does a NEW, so it will wipe out any resident Applesoft program. Also note that the mini-assembler loads from $800 to $947. Don't try to assemble anything into those locatiions.

```
0000:              2 ************************
0000:              3 *                      *
0000:              4 *       APPLE II       *
0000:              5 *    MINI-ASSEMBLER    *
0000:              6 *                      *
0000:              7 ************************
002E:              8 FORMAT  EQU  $2E
002F:              9 LENGTH  EQU  $2F
0031:             10 MODE    EQU  $31
0033:             11 PROMPT  EQU  $33
0034:             12 YSAV    EQU  $34
0035:             13 L       EQU  $35
003A:             14 PCL     EQU  $3A
003B:             15 PCH     EQU  $3B
003D:             16 A1H     EQU  $3D
003E:             17 A2L     EQU  $3E
003F:             18 A2H     EQU  $3F
0042:             19 A4L     EQU  $42
0043:             20 A4H     EQU  $43
0044:             21 FMT     EQU  $44
0200:             22 IN      EQU  $200
D64B:             23 NEW     EQU  $D64B
F88E:             24 INSDS2  EQU  $F88E
F8D0:             25 INSTDSP EQU  $F8D0
F94A:             26 PRBL2   EQU  $F94A
F953:             27 PCADJ   EQU  $F953
F9B4:             28 CHAR1   EQU  $F9B4
F9BA:             29 CHAR2   EQU  $F9BA
F9C0:             30 MNEML   EQU  $F9C0
FA00:             31 MNEMR   EQU  $FA00
FC1A:             32 CURSUP  EQU  $FC1A
FD67:             33 GETLNZ  EQU  $FD67
FDED:             34 COUT    EQU  $FDED
FE00:             35 BL1     EQU  $FE00
FE78:             36 A1PCLP  EQU  $FE78
FF3A:             37 BELL    EQU  $FF3A
FFA7:             38 GETNUM  EQU  $FFA7
FFBE:             39 TOSUB   EQU  $FFBE
FFC7:             40 ZMODE   EQU  $FFC7
FFCC:             41 CHRTBL  EQU  $FFCC
----- NEXT OBJECT FILE NAME IS MINI-ASSEMBLER.OBJ0
0800:             42         ORG  $800
0800: 20 4B D6    43         JSR  NEW        ;SCRATCH ANY BASIC PROGRAM
0803: A9 00       44         LDA  #0
0805: 8D 00 08    45         STA  $800
0808: 4C 9D 08    46         JMP  RESETZ     ;BRANCH TO START OF MINI/ASSEM
080B: E9 81       47 REL     SBC  #$81       ;IS FMT COMPATIBLE
080D: 4A          48         LSR  A          ;WITH RELATIVE MODE?
080E: D0 14       49         BNE  ERR3       ;NO.
0810: A4 3F       50         LDY  A2H
0812: A6 3E       51         LDX  A2L        ;DOUBLE DECREMENT
0814: D0 01       52         BNE  REL2
0816: 88          53         DEY
0817: CA          54 REL2    DEX
0818: 8A          55         TXA
0819: 18          56         CLC
081A: E5 3A       57         SBC  PCL        ;FORM ADDRESS PC-2
081C: 85 3E       58         STA  A2L
```

```
081E: 10 01      59          BPL   REL3
0820: C8         60          INY
0821: 98         61 REL3     TYA
0822: E5 3B      62          SBC   PCH
0824: D0 6B      63 ERR3     BNE   ERR         ;TO FAR TO BRANCH ERROR
0826:            64 *
0826: A4 2F      65 FINDOP   LDY   LENGTH
0828: B9 3D 00   66 FNDOP2   LDA   A1H,Y       ;MOVE INST TO (PC)
082B: 91 3A      67          STA   (PCL),Y
082D: 88         68          DEY
082E: 10 F8      69          BPL   FNDOP2
0830: 20 1A FC   70          JSR   CURSUP
0833: 20 1A FC   71          JSR   CURSUP      ;RESTORE CURSOR
0836: 20 D0 F8   72          JSR   INSTDSP     ;TYPE FORMATTED LINE
0839: 20 53 F9   73          JSR   PCADJ       ;UPDATE PC
083C: 84 3B      74          STY   PCH
083E: 85 3A      75          STA   PCL
0840: 4C A0 08   76          JMP   NXTLINE     ;GET NEXT LINE
0843: 20 BE FF   77 FAKEMON3 JSR   TOSUB       ;GO TO DELIM HANDLER
0846: A4 34      78          LDY   YSAV        ;RESTORE Y-INDEX
0848: 20 A7 FF   79 FAKEMON  JSR   GETNUM      ;READ PARAM
084B: 84 34      80          STY   YSAV        ;SAVE Y-INDEX
084D: A0 17      81          LDY   #$17        ;INIT DELIM INDEX
084F: 88         82 FAKEMON2 DEY               ;CHECK NEXT DELIM
0850: 30 4B      83          BMI   RESETZ      ;ERR IF UNRECOGNIZED DELIM
0852: D9 CC FF   84          CMP   CHRTBL,Y    ;COMPARE WITH DELIM TABLE
0855: D0 F8      85          BNE   FAKEMON2    ;NO MATCH
0857: C0 15      86          CPY   #$15        ;MATCH, IS IT CR?
0859: D0 E8      87          BNE   FAKEMON3    ;NO HANDLE IT IN MONITOR
085B: A5 31      88          LDA   MODE
085D: A0 00      89          LDY   #0
085F: C6 34      90          DEC   YSAV
0861: 20 00 FE   91          JSR   BL1         ;HANDLE CR OUTSIDE MONITOR
0864: 4C A0 08   92          JMP   NXTLINE
0867:            93 *
0867: A5 3D      94 TRYNEXT  LDA   A1H         ;GET TRIAL OPCODE
0869: 20 8E F8   95          JSR   INSDS2      ;GET FMT+LENGTH FOR OPCODE
086C: AA         96          TAX
086D: BD 00 FA   97          LDA   MNEMR,X     ;GET LOWER MNEMONIC BYTE
0870: C5 42      98          CMP   A4L         ;MATCH ?
0872: D0 13      99          BNE   NEXTOP      ;NO, TRY NEXT OPCODE
0874: BD C0 F9  100          LDA   MNEML,X     ;GET UPPER MNEMONIC BYTE
0877: C5 43     101          CMP   A4H         ;MATCH ?
0879: D0 0C     102          BNE   NEXTOP      ;NO, TRY NEXT OPCODE
087B: A5 44     103          LDA   FMT
087D: A4 2E     104          LDY   FORMAT      ;GET TRIAL FORMAT
087F: C0 9D     105          CPY   #$9D        ;IS TRIAL FORMAT RELATIVE?
0881: F0 88     106          BEQ   REL         ;YES
0883: C5 2E     107          CMP   FORMAT      ;SAME FORMAT?
0885: F0 9F     108          BEQ   FINDOP      ;YES
0887: C6 3D     109 NEXTOP  DEC   A1H         ;NO, TRY NEXT OPCODE
0889: D0 DC     110          BNE   TRYNEXT
088B: E6 44     111          INC   FMT         ;NO MORE, TRY WITH LEN=2
088D: C6 35     112          DEC   L           ;WAS L=2 ALREADY?
088F: F0 D6     113          BEQ   TRYNEXT     ;NO
0891:           114 *
0891: A4 34     115 ERR     LDY   YSAV        ;YES, UNRECOGNIZED INST.
0893: 98        116 ERR2    TYA
```

```
0894: AA          117           TAX
0895: 20 4A F9    118           JSR   PRBL2      ;PRINT ^ UNDER LAST READ
0898: A9 DE       119           LDA   #$DE       ;CHAR TO INDICATE ERROR
089A: 20 ED FD    120           JSR   COUT       ;POSITION.
089D: 20 3A FF    121 RESETZ    JSR   BELL
08A0: A9 A1       122 NXTLINE   LDA   #$A1       ;'!'
08A2: 85 33       123           STA   PROMPT     ;INITIALIZE PROMPT
08A4: 20 67 FD    124           JSR   GETLNZ     ;GET LINE
08A7: 20 C7 FF    125           JSR   ZMODE      ;INIT SCREEN STUFF
08AA: AD 00 02    126           LDA   IN         ;GET CHAR
08AD: C9 A0       127           CMP   #$A0       ;ASCII SPACE?
08AF: F0 13       128           BEQ   SPACE      ;YES
08B1: C8          129           INY
08B2: C9 A4       130           CMP   #$A4       ;ASCII '$' IN COL 1?
08B4: F0 92       131           BEQ   FAKEMON    ;YES, SIMULATE MONITOR
08B6: 88          132           DEY              ;NO, BACKUP A CHAR
08B7: 20 A7 FF    133           JSR   GETNUM     ;GET A NUMBER
08BA: C9 93       134           CMP   #$93       ;':' TERMINATOR?
08BC: D0 D5       135 ERR4      BNE   ERR2       ;NO, ERR
08BE: 8A          136           TXA
08BF: F0 D2       137           BEQ   ERR2       ;NO ADDR PRECEDING COLON
08C1: 20 78 FE    138           JSR   A1PCLP     ;MOVE ADDR TO PCL,PCH
08C4: A9 03       139 SPACE     LDA   #$03       ;COUNT OF CHAR IN MNEMONIC
08C6: 85 3D       140           STA   A1H
08C8: 20 3F 09    141 NXTMN     JSR   GETNSP     ;GET 1ST MNEMONIC CHARACTER
08CB: 0A          142           ASL   A
08CC: E9 BE       143           SBC   #$BE       ;SUBTRACT OFFSET
08CE: C9 C2       144           CMP   #$C2       ;LEGAL CHARACTER?
08D0: 90 C1       145           BCC   ERR2       ;NO
08D2: 0A          146           ASL   A          ;COMPRESS-LEFT JUSTIFY
08D3: 0A          147           ASL   A
08D4: A2 04       148           LDX   #$04
08D6: 0A          149 NXTM2     ASL   A          ;DO 5 TRIPPLE WORD SHIFTS
08D7: 26 42       150           ROL   A4L
08D9: 26 43       151           ROL   A4H
08DB: CA          152           DEX
08DC: 10 F8       153           BPL   NXTM2
08DE: C6 3D       154           DEC   A1H        ;DONE WITH 3 CHAR?
08E0: F0 F4       155           BEQ   NXTM2      ;YES BUT DO 1 MORE SHIFT
08E2: 10 E4       156           BPL   NXTMN      ;NO
08E4:             157 *
08E4: A2 05       158           LDX   #$05       ;5 CHARACTER ADDRESSING MODE
08E6: 20 3F 09    159 FORM2     JSR   GETNSP     ;GET 1ST CHARACTER OF ADDRESS
08E9: 84 34       160           STY   YSAV
08EB: DD B4 F9    161           CMP   CHAR1,X    ;1ST CHARACTER MATCH PATTERN ?
08EE: D0 13       162           BNE   FORM3      ;NO
08F0: 20 3F 09    163           JSR   GETNSP     ;YES, GET 2ND CHAR
08F3: DD BA F9    164           CMP   CHAR2,X    ;MATCHES SECOND HALF?
08F6: F0 0D       165           BEQ   FORM5      ;YES
08F8: BD BA F9    166           LDA   CHAR2,X    ;NO, IS 2ND HALF ZERO?
08FB: F0 07       167           BEQ   FORM4      ;YES
08FD: C9 A4       168           CMP   #$A4       ;NO 2ND HAL OPTIONAL?
08FF: F0 03       169           BEQ   FORM4      ;YES
0901: A4 34       170           LDY   YSAV
0903: 18          171 FORM3     CLC              ;CLEAR CARRY NO MATCH
0904: 88          172 FORM4     DEY              ;BACK UP ONE CHARACTER
0905: 26 44       173 FORM5     ROL   FMT        ;FORM FORMAT BYTE
0907: E0 03       174           CPX   #$03       ;TIME TO CHECK FOR ADDRESS
```

```
0909:D0 0D      175           BNE  FORM7      ;NO
090B:20 A7 FF   176           JSR  GETNUM     ;YES
090E:A5 3F      177           LDA  A2H
0910:F0 01      178           BEQ  FORM6      ;HIGH ORDER BYTE ZERO
0912:E8         179           INX             ;NO, INCR FOR 2-BYTE
0913:86 35      180 FORM6     STX  L          ;STORE LENGTH
0915:A2 03      181           LDX  #$03       ;RELOAD FORMAT INDEX
0917:88         182           DEY             ;BACKUP A CHARACTER
0918:86 3D      183 FORM7     STX  A1H        ;SAVE INDEX
091A:CA         184           DEX             ;DONE WITH FORMAT CHECK ?
091B:10 C9      185           BPL  FORM2      ;NO
091D:A5 44      186           LDA  FMT        ;YES PUT LENGTH
091F:0A         187           ASL  A          ;IN LOW BITS
0920:0A         188           ASL  A
0921:05 35      189           ORA  L
0923:C9 20      190           CMP  #$20
0925:B0 06      191           BCS  FORM8      ;ADD $ IF NONZERO LENGTH
0927:A6 35      192           LDX  L          ;AND DON'T ALREADY HAVE IT
0929:F0 02      193           BEQ  FORM8
092B:09 80      194           ORA  #$80
092D:85 44      195 FORM8     STA  FMT
092F:84 34      196           STY  YSAV
0931:B9 00 02   197           LDA  IN,Y       ;GET NEXT NONBLANK
0934:C9 BB      198           CMP  #$BB       ;';' START OF COMMENT
0936:F0 04      199           BEQ  FORM9      ;YES
0938:C9 8D      200           CMP  #$8D       ;A CARRAIGE RETURN?
093A:D0 80      201           BNE  ERR4       ;NO, ERR
093C:4C 67 08   202 FORM9     JMP  TRYNEXT
093F:           203 *
093F:B9 00 02   204 GETNSP    LDA  IN,Y
0942:C8         205           INY
0943:C9 A0      206           CMP  #$A0       ;GET NEXT NON BLANK CHAR
0945:F0 F8      207           BEQ  GETNSP
0947:60         208           RTS
0948:           209 *

*** SUCCESSFUL ASSEMBLY:  NO ERRORS
```

THE APPLE II CASSETTE INTERFACE

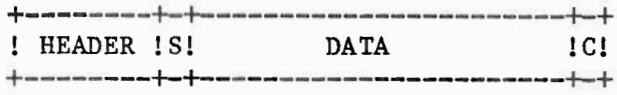
INTRODUCTION:

This note is to explain the cassette interface built into the Apple II and Apple II plus. Included is information on the format and use of the read and write subroutines. It is assumed in this note that the cassette recorder is in the proper mode, play or record, when the read and write routines are executed. Also note that the timing is approximate and may vary from one Apple to another.

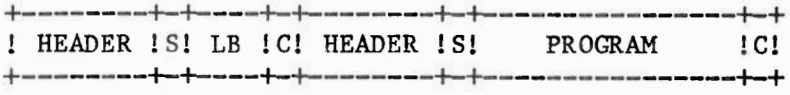
RECORDS:

A record is a block of binary data. This data may be a BASIC or APPLESOFT program, a machine language program, or just binary data. Records representing BASIC or APPLESOFT programs are really two records, the length of the program and the actual program. A record consists of a header, sync bit, the actual data, and a checksum byte for error detection.

MONITOR RECORD FORMAT



BASIC PROGRAM RECORD FORMAT

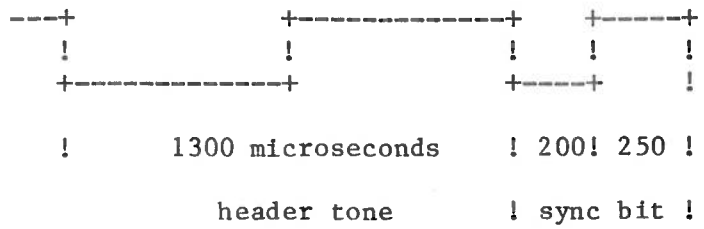


KEY: S = SYNC BIT
C = CHECKSUM BYTE
LB = BASIC PROGRAM LENGTH

THE HEADER:

The header consists of 10 seconds of 770 Hz tone, (1 cycle equals 1300 microseconds). This is enough time for the cassette motor to get up to speed and the plastic tape leader to go by. There is also a shortened header between the BASIC length bytes and the BASIC program itself. This header is generated by a subroutine called HEADR. The value of the accumulator on entry controls the length of the header tone. This can vary from 0.2 seconds to 40 seconds. On entry the X register should be 0 and the carry flag should be set. HEADR also generates a sync bit at the end of the tone. HEADR resides at hexadecimal address $FCC9, or decimal address -882.

THE LAST CYCLE OF HEADER TONE AND SYNC BIT



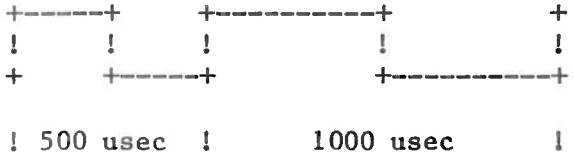
THE SYNC BIT:

The sync bit is one half cycle of 2500 Hz, (200 microseconds) and one half cycle of 2000 Hz, (250 microseconds). It is used to signal the end of the header tone and the start of the data. It is generated by HEADR.

THE DATA:

The data is recorded on the tape with a low starting address and a high ending address. Each byte of data is shifted out most significant bit first, least significant bit last. A zero bit is made up of one cycle of 2 kHz, (250 microseconds per half cycle) and a one bit is one cycle of 1 kHz, (500 microseconds per half cycle). This works out to 2000 baud for zeros only and 1000 baud for ones, an average of 1500 baud.

A ZERO BIT AND A ONE BIT



THE CHECKSUM:

All during reading or writing each data byte is EXCLUSIVE OR-ed with a checksum byte. This byte is writen on the tape at the end of the data block. If the checksum computed during a read agrees with the checksum that was written out then the data is probably good. This method will detect an odd number of errors for any of the eight bits of the byte.

WRITING DATA:

The cassette output circuitry is quite simple. it is a flip-flop connected through a voltage divider to the jack on the back panel of the Apple. Any time the address $C020 is accessed this flip-flop changes state. Accessing the flip-flop once every 500 microseconds generates a 1000 Hz tone.

READING DATA:

The cassette input circuit more complicated. It consists of a 741 operational amplifier configured as a zero crossing detector. That means that whenever the voltage at the input jack goes from positive to negative (or negative to positive) the output of the amplifier switches from a 1 to a 0 (or 0 to 1). The detector is accessed by any read to address $C060. The sign bit (most significant bit) reflects the detector status. The read routines continually EXCLUSIVE ORs this bit with the value most recently read to detect a change in state. The amount of time required to change state indicates the incoming frequency which then is used to determine if a one or a zero has been recieved. After detecting the first zero crossing at the start of a read, the read routine uses HEADR to generate a 3.5 delay then waits for the sync bit. It then reads the data and puts it in the specified memory range.

USING THE CASSETTE INTERFACE:

To either read or write all you need do is specify an address range and execute the read or write subroutine. The address range is stored in four bytes, two for the start and two for the end. In both cases the least significant byte is first.

FROM THE MONITOR:

If the start is $800 and the end is $9FF then

800.9FFW will write the data to the cassette and
800.9FFR will retrieve it.

FROM MACHINE LANGUAGE:

Again if the start is $800 and the end is $9FF then store the address range,

```
LDA #$00
STA $3C     starting address low
LDA #$08
STA $3D     starting address high
LDA #$FF
STA $3E     ending address low
LDA #$09
STA $3F     ending address high
```

then JSR $FEDC to write the data to the cassette or JSR $FEFD to read from the cassette.

FROM BASIC:

First set up the address range. If S = the start and E = the end then from integer BASIC,

```
POKE 60,S MOD 256
POKE 61,S / 256
POKE 62,E MOD 256
POKE 63,E / 256
```

or from APPLESOFT,

```
POKE 60,S - INT(S / 256) * 256
POKE 61,S / 256
POKE 62,E - INT(E / 256) * 256
POKE 63,E / 256
```

Then to write out to cassette CALL -307 or to read in from the cassette CALL -259.

SERIAL HANDSHAKE MODIFICATION
22 AUG 79

INTRODUCTION:

The High Speed Serial Interface card cannot run faster than 300 baud on most printers due to a lack of a printer busy line. This modification uses the existing data input line to sense if the printer is busy and inhibit output if necessary. DOS 3.2 is required to use this modification.

WARNING:

Damage to the High Speed Serial Interface card may not be covered by your warranty. If you aren't sure of the signal levels and pinout of your printer, find out or get someone who knows to help you.

WIRING CHANGES:

First you must determine which wire your printer uses to indicate a printer busy or buffer full condition. Your printer's manual should contain this information or contact the manufacturer.

Examples:

IDS 125/225	pin 4
HEATH H-14	pin 4
TI-810	pin 11
SPINTERM	pin 19
COMPRINT	pin 20

The preferred place to do the wiring change is in the cable, but it can also be done at the Serial Card or the printer. Disconnect the wire between pin 2 of the printer and pin 2 on the Serial Card. Then connect the wire with the printer busy signal to the wire for pin 2 on the Serial Card.

SOFTWARE PATCH:

Next you must decide which slot the interface will go in and type in the software patch. The patch is customized for this slot number and won't work if used with a different configuration. The patch forces the computer to look to see if the printer is busy and wait if it is. Enter the patch using the values from the table for words in brackets, < >.

SLOT	1	2	3	4	5	6	7
DATA	90	A0	B0	C0	D0	E0	F0
CODE	C1	C2	C3	C4	C5	C6	C7

Enter the monitor with CALL -155 and type

```
3B0:A9 <SLOT>
:20 95 FE
:A9 00
:20 ED FD
:A9 C5
:85 36
:A9 03
:85 37
:4C EA 03
:2C <DATA> C0
:30 FB
:4C 07 <CODE>
:00 00 00
```

To check your typing, type
3B0L
and compare your listing to the one below for slot 1.

```
03B0-   A9 01       LDA   #$01
03B2-   20 95 FE    JSR   $FE95
03B5-   A9 00       LDA   #$00
03B7-   20 ED FD    JSR   $FDED
03BA-   A9 C5       LDA   #$C5
03BC-   85 36       STA   $36
03BE-   A9 03       LDA   #$03
03C0-   85 37       STA   $37
03C2-   4C EA 03    JMP   $03EA
03C5-   2C 90 C0    BIT   $C090
03C8-   30 FB       BMI   $03C5
03CA-   4C 07 C1    JMP   $C107
03CB-   00          BRK
03CC-   00          BRK
03CD-   00          BRK
```

Now return to basic with 3D0G.

SAVING THE PATCH TO DISK:

The patch must be in memory before the printer can be used at the higher speeds. Save the patch by typing
BSAVE PATCH, A$3B0, L$20.

USING THE PRINTER:

The first time you want to use the printer you must load the patch and initialize the interface. From immediate mode type
BLOAD PATCH
CALL 944.

This may be done in a program,by entering
100 PRINT D$;"BLOAD PATCH": CALL 944
assuming that D$ is a control-D.

If in the course of the program you need to turn off the printer, type
PR#0
or in a program enter
200 PRINT D$;"PR#0"

Then to reconnect the printer, all that is required is
CALL 954 or from a program
300 CALL 954.

NOTES:

If the printer does not print after the CALL 944, it is probably sending the opposite polarity busy signal. The patch can be changed to recognize with
POKE 968,16.
If this doesn't work, have the printer checked.

The modification allows the speed, column width, and other variables to be changed with the POKEs in the manual.

SERIAL HANDSHAKE MODIFICATION FOR PASCAL
JUNE 1980

INTRODUCTION:

The High Speed Serial Interface card will over-run the buffer on most printers at speeds greater than 300 baud due to it's lack of a busy line. This note shows how to use the existing data input line to stop output while the printer is busy. The modification will not affect other Serial cards or other cards in slot 1. 13 bytes of the user area of BIOS starting at $DAC0 are used by this modification.

WARNING:

Damage to the High Speed Serial Interface card while installing this modification will not be covered by your warranty. If you aren't sure of the signal levels and pinout of your printer, find out or ask your Service Center to help you.

WIRING CHANGES:

First you must determine which wire your printer uses to indicate a printer busy or buffer full condition. Your printer's manual should contain this information or contact the manufacturer.

Examples:

IDS 125/225	pin 4
HEATH H-14	pin 4
TI-810	pin 11
SPINTERM	pin 19
COMPRINT	pin 20

The preferred place to do the wiring change is in the cable, but it can also be done at the Serial Card or the printer. Disconnect the wire between pin 2 of the printer and pin 2 on the Serial Card. Then connect the wire with the printer busy signal to the wire for pin 2 on the Serial Card.

SOFTWARE CHANGES:

The following program will change the SYSTEM.APPLE on the default diskette to utilize the above hardware modification. Since this modifies one of the SYSTEM files, be sure that you have a backup diskette to protect you from errors.

```
PROGRAM HANDSHAKE;

VAR
   BLK:PACKED ARRAY [0..511] OF 0..255;
   BLT:INTEGER;
   F  :FILE;
   FN :STRING;

BEGIN
  FN := 'SYSTEM.APPLE';
  RESET(F,FN);
  BLT:=BLOCKREAD(F,BLK,1,3);
  BLK[504]:=192;
  BLK[505]:=218;
  BLT:=BLOCKWRITE(F,BLK,1,3);
  BLT:=BLOCKREAD(F,BLK,1,5);
  BLK[192]:=32;   (* JSR $D681     CHECK CONSOLE *)
  BLK[193]:=129;
  BLK[194]:=214;
  BLK[195]:=192;  (* CPY #$10      SLOT 1? *)
  BLK[196]:=16;
  BLK[197]:=208;  (* BNE +$05      NO, DON'T CHECK *)
  BLK[198]:=5;
  BLK[199]:=185;  (* LDA $C080,Y   CHECK FOR BUSY *)
  BLK[200]:=128;
  BLK[201]:=192;
  BLK[202]:=16;   (* BPL -$0C      BUSY, GO CHECK CONSOLE *)
  BLK[203]:=244;
  BLK[204]:=96;   (* RTS           OK TO PRINT NOW *)
  BLT:=BLOCKWRITE(F,BLK,1,5);
  CLOSE(F,LOCK)
END.
```

Now Quit the EDITOR and R)un the program, assuming that you use the work file. Then turn the Apple off and back on. The patch is now installed and working. To check it, enter the FILER, ask for a directory L)ist, and reply to DIR LIST OF? with

```
#4,PRINTER:
```

This should give you a directory listing of the diskette in device #4 on the printer. If nothing happens then change:

```
BLK[202]:=16;
```

to

```
BLK[202]:=48;
```

Then re-compile and check it again. If it still doesn't print then there's a hardware problem. Check the wiring change or ask your Service Center to check the interface and the printer.

TTY DRIVER ROUTINE - REV. 1

This program is intended to allow the TTY Driver description in the Apple II Reference Manual to be used with many low speed serial printers. This program has the following capabilities:

1. Each carriage return is followed by a line feed and line feed delay.

2. A form of TAB function (HTAB in Applesoft) is available.

3. BASIC listings may be formatted for wide column printers.

4. Comma (,) functions will not execute normally in most cases.

To load the program in the Apple, use the attached disassembler listing and enter the code either in HEX or the mini-assembler. Once loaded, the code may be saved (in the monitor) with:

```
* 300.3AFW
```

To reload the tape, type:

```
* 300.AFR
```

To save on disk be sure the disk is booted before loading the program. Then type:

```
* BSAVE (File Name), A$300, L$AF
```

To load from disk, simply type

```
> BLOAD (File Name)
```

You may find it convenient to load this program with the HELLO program on disk.

To activate the print routine use

```
CALL 771 (or *303G)
```

for operation without DOS or

CALL 773 (or *305G)

for operation with DOS. (Be sure DOS has been booted)

The system will now provide output properly to the printer but not to the screen.

For formatted BASIC listings beyond 40 char:

POKE 769, (Printer column width) (*301: (printer width))

To execute a TAB use:

POKE 36, (tab) (*24: (tab))

For 300 BAUD change location $38F to $59 and location $392 to $10. For other BAUD rates use the attached calculation information.

NOTE: If you are using DOS 3.2, then you must modify the routine. From BASIC type:

POKE 800,110

and save the program again.

```
]ASM
                        2 *
                        3 *  TTYDRIVER
                        4 *     ROUTINE
                        5 *       REV 1
                        6 * COPYRIGHT 1978 BY
                        7 * APPLE COMPUTER INC.
                        8 *    8/15/78
                        9 *  R WIGGINTON
                       10 *  W SANDER
                       12        ORG   $300
                       13        OBJ   $4300
                       14 CH     EQU   $24   ;CURSOR HORIZONTAL
                       15 CSWL   EQU   $36   ;CHAR OUT SWITCH
                       16 CSWH   EQU   $37
                       17 WAIT   EQU   $FCA8
                       18 ZTEMPL EQU   $2A
                       19 ZTEMPH EQU   $2B
                       20 STATUS EQU   $C08E
                       21 FLAGS  EQU   $778
                       22 STAT   EQU   $7F8
                       23 MARK   EQU   $C058
                       24 SPACE  EQU   $C059
                       25 YSAVE  DS    1
                       26 CWIDTH DS    1   ;COLUMN WIDTH
                       27 COLCNT DS    1   ;CURRENT COLUMN COUNT
0303 18                28 TTINT  CLC
0304 B0                29        HEX   B0  ;BCS
0305 38                30        HEX   38  ;SEC
0306 98                31        TYA   ;SAVE Y-REG
0307 48                32        PHA
0308 08                33        PHP
0309 A9 28             34        LDA   #$28
030B 8D 01 03          35        STA   CWIDTH
030E A9 36             36        LDA   #$36
0310 85 2A             37        STA   ZTEMPL
0312 A0 00             38        LDY   #$0
0314 8C 02 03          39        STY   COLCNT
0317 84 2B             40        STY   ZTEMPH
0319 28                41        PLP
031A 90 0E             42        BCC   NODOS   ;BRANCH IF NOT SETTING LOS
031C AD E7 03          43        LDA   $3E7
031F E9 6D             44        SBC   #$6D  ;SET DOS COUT POINTERS
0321 85 2A             45        STA   ZTEMPL
0323 AD E8 03          46        LDA   $3E8
0326 E9 00             47        SBC   #$0
0328 85 2B             48        STA   ZTEMPH
032A A9 36             49 NODOS  LDA   #<TTOUT
032C 91 2A             50        STA   (ZTEMPL),Y
032E C8                51        INY
032F A9 03             52        LDA   #>TTOUT
0331 91 2A             53        STA   (ZTEMPL),Y
0333 68                54        PLA   ;RECOVER Y:REG
0334 A8                55        TAY
0335 60                56 RTS1   RTS
0336 48                57 TTOUT  PHA   ;SAVE TWICE ON STACK
0337 48                58        PHA
0338 AD 02 03          59 TTOUT2 LDA   COLCNT
033B C5 24             60        CMP   CH        ;CHECK FOR TAB
033D 68                61        PLA   ;RESTORE CHARACTER
```

```
033E B0 03       62          BCS   TESTCTRL  ;IF C SET, NO TAB
0340 48          63          PHA
0341 A9 A0       64          LDA   #$A0      ;PRINT A SPACE
0343 2C 35 03    65 TESTCTRL BIT    RTS1      TEST FOR CTRL CHAR
0346 F0 03       66          BEQ   PRINTIT
0348 EE 02 03    67          INC   COLCNT    ;IF NOT CTRL CHAR
034B 08          68 PRINTIT PHP
034C 20 7C 03    69          JSR   DOCHAR    ;OUTPUT THE CHAR
034F 28          70          PLP
0350 68          71          PLA   ;RESTORE CHAR
0351 48          72          PHA   ;AND PUT BACK ON STACK
0352 90 E4       73          BCC   TTOUT2    ;CHECK FOR MORE SPACES
0354 49 0D       74          EOR   #$0D      ;CHECK FOR CAR RETURN
0356 0A          75          ASL
0357 D0 0D       76          BNE   FINISH    ;IF NOT CAR RET, DONE
0359 8D 02 03    77          STA   COLCNT    ;CLEAR COLUMN COUNT
035C A9 8A       78          LDA   #$8A      ;DO LINE FEED
035E 20 7C 03    79          JSR   DOCHAR
0361 A9 58       80          LDA   #$58      ;200MSEC DELAY
0363 20 A8 FC    81          JSR   WAIT      ;AFTER LINE FEED
0366 A9 00       82 FINISH LDA   #$0       ;ZERO CURSOR HORIZ
0368 85 24       83          STA   CH
036A AD 02 03    84 FINISH1 LDA   COLCNT
036D F0 09       85          BEQ   SETCH     ;RESET CH IF CAR RETURN
036F ED 01 03    86          SBC   CWIDTH    ;CHECK IF IN MARGIN
0372 E9 F7       87          SBC   #$F7      ;FOR CAR RETURN
0374 90 04       88          BCC   RETURN    ;IF NOT ,FINISHED
0376 69 1F       89          ADC   #$1F
0378 85 24       90 SETCH   STA   CH
037A 68          91 RETURN PLA
037B 60          92          RTS   ;RETURN TO CALLER
037C 8C 00 03    93 DOCHAR STY   YSAVE
037F 08          94          PHP   ;SAVE STATUS
0380 A0 0B       95          LDY   #$0B      ;11 BITS (1 START, 8 DATA, 2 STOP)
0382 18          96          CLC   ;BEGIN WITH SPACE
0383 48          97 TTOUT3 PHA   ;SAVE A-REG
0384 B0 05       98          BCS   MARKOUT
0386 AD 59 C0    99          LDA   SPACE     ;SEND A SPACE
0389 90 03      100          BCC   TTOUT4
038B AD 58 C0   101 MARKOUT LDA   MARK    ;SEND A MARK
038E A9 D7      102 TTOUT4 LDA   #$D7      ;DELAY 9.091MSEC
0390 48         103 DLY1   PHA   ;FOR 110 BAUD
0391 A9 20      104          LDA   #$20
0393 4A         105 DLY2   LSR
0394 90 FD      106          BCC   DLY2
0396 68         107          PLA
0397 E9 01      108          SBC   #$1
0399 D0 F5      109          BNE   DLY1
039B 68         110          PLA
039C 6A         111          ROR   ;NEXT BIT
039D 88         112          DEY
039E D0 E3      113          BNE   TTOUT3
03A0 AC 00 03   114          LDY   YSAVE     ;RESTORE Y-REG
03A3 28         115          PLP;R STATUS
03A4 60         116          RTS   ;RETURN

--- END ASSEMBLY ---

TOTAL ERRORS: 00
```

```
6
*300LLLLL

0300-   00          BRK
0301-   7F          ???
0302-   7F          ???
0303-   18          CLC
0304-   B0 38       BCS   $033E
0306-   98          TYA
0307-   48          PHA
0308-   08          PHP
0309-   A9 28       LDA   #$28
030B-   8D 01 03    STA   $0301
030E-   A9 36       LDA   #$36
0310-   85 2A       STA   $2A
0312-   A0 00       LDY   #$00
0314-   8C 02 03    STY   $0302
0317-   84 2B       STY   $2B
0319-   28          PLP
031A-   90 0E       BCC   $032A
031C-   AD E7 03    LDA   $03E7
031F-   E9 6D       SBC   #$6D
0321-   85 2A       STA   $2A
0323-   AD E8 03    LDA   $03E8
0326-   E9 00       SBC   #$00
0328-   85 2B       STA   $2B
032A-   A9 36       LDA   #$36
032C-   91 2A       STA   ($2A),Y
032E-   C8          INY
032F-   A9 03       LDA   #$03
0331-   91 2A       STA   ($2A),Y
0333-   68          PLA
0334-   A8          TAY
0335-   60          RTS
0336-   48          PHA
0337-   48          PHA
0338-   AD 02 03    LDA   $0302
033B-   C5 24       CMP   $24
033D-   68          PLA
033E-   B0 03       BCS   $0343
0340-   48          PHA
0341-   A9 A0       LDA   #$A0
0343-   2C 35 03    BIT   $0335
0346-   F0 03       BEQ   $034B
0348-   EE 02 03    INC   $0302
034B-   08          PHP
034C-   20 7C 03    JSR   $037C
034F-   28          PLP
0350-   68          PLA
0351-   48          PHA
0352-   90 E4       BCC   $0338
0354-   49 0D       EOR   #$0D
0356-   0A          ASL
0357-   D0 0D       BNE   $0366
0359-   8D 02 03    STA   $0302
035C-   A9 8A       LDA   #$8A
035E-   20 7C 03    JSR   $037C
0361-   A9 58       LDA   #$58
0363-   20 A8 FC    JSR   $FCA8
0366-   A9 00       LDA   #$00
0368-   85 24       STA   $24
036A-   AD 02 03    LDA   $0302
036D-   F0 09       BEQ   $0378
```

```
037F-  ED 01 03    SBC   $0301
0382-  E9 F7       SBC   #$F7
0384-  90 04       BCC   $038A
0386-  69 1F       ADC   #$1F
0388-  85 24       STA   $24
038A-  68          PLA
038B-  60          RTS
038C-  8C 00 03    STY   $0300
038F-  08          PHP
0390-  A0 0B       LDY   #$0B
0392-  18          CLC
0393-  48          PHA
0394-  B0 05       BCS   $039B
0396-  AD 59 C0    LDA   $C059
0399-  90 03       BCC   $039E
039B-  AD 58 C0    LDA   $C058
039E-  A9 D7       LDA   #$D7
03A0-  48          PHA
03A1-  A9 20       LDA   #$20
03A3-  4A          LSR
03A4-  90 FD       BCC   $03A3
03A6-  68          PLA
03A7-  E9 01       SBC   #$01
03A9-  D0 F5       BNE   $03A0
03AB-  68          PLA
03AC-  6A          ROR
03AD-  88          DEY
03AE-  D0 E3       BNE   $0393
03B0-  AC 00 03    LDY   $0300
03B3-  28          PLP
03B4-  60          RTS
03B5-  00          BRK
03B6-  00          BRK
03B7-  00          BRK
03B8-  00          BRK
```

TTY DRIVER BAUD CALCULATION

FORMULA

1. Determine Divisor D to nearest integer

$$D = \frac{14318180 * 65}{912 * (\text{Baud Rate})}$$

2. For selected 'N' determine 'M'
   'M' must be a hex integer between $0 and $FF

BYTE AT $3D7 ($3C6 in 40) col + vid version N	BYTE AT $3D4 ($3C3 in 40) col + vid version M	ACTUAL BAUD RATE For chosen M
$80	(D-23)/53	F/(53*M + 23)
$40	(D-23)/48	F/(48*M + 23)
$20	(D-23)/43	F/(43*M + 23)
$10	(D-23)/38	F/(38*M + 23)
$08	(D-23)/33	F/(33*M + 23)
$04	(D-23)/28	F/(28*M + 23)
$02	(D-23)/23	F/(23*M + 23)
		$F = \frac{14318180 * 65}{912}$ $F = 1.0204843 \times 10^6$

For 110, D = 9277, M = $D7 = 215 B = 110.2 Hz N = $20

For 300, D = 34016 M = $59 = 89 B = 299.7 Hz N = $10

CENTRONICS 730 DRIVER
4 OCT 79

INTRODUCTION:

This driver allows the user to select expanded or normal print mode with a poke. DOS 3.2 is required to use this routine.

SOFTWARE ENTRY

First you must decide which slot the interface will go in and enter the driver. The routine is customized for this slot number and won't work properly if used with a different configuration. Enter the driver using the values from the table for words in brackets, < >.

SLOT	1	2	3	4	5	6	7
CODE	C1	C2	C3	C4	C5	C6	C7

Enter the monitor with CALL -155 and type

```
3B0:A9 <SLOT>
:20 95 FE
:A9 80
:20 ED FD
:A9 C5
:85 36
:A9 03
:85 37
:4C EA 03
:29 7F
:0D CD 03
:4C 02 <CODE>
:80
```

To check your typing, type
3B0L
and compare your listing to the one below for slot 1.

```
03B0-   A9 01       LDA   #$01
03B2-   20 95 FE    JSR   $FE95
03B5-   A9 80       LDA   #$80
03B7-   20 ED FD    JSR   $FDED
03BA-   A9 C5       LDA   #$C5
03BC-   85 36       STA   $36
03BE-   A9 03       LDA   #$03
03C0-   85 37       STA   $37
03C2-   4C EA 03    JMP   $03EA
03C5-   29 7F       AND   #$7F
03C7-   0D CD 03    ORA   $03CD
03CA-   4C 02 C1    JMP   $C102
03CD-   80          ???
```

Now return to BASIC with 3D0G

SAVING THE PROGRAM TO DISK:

The driver should be in memory before the printer is used.
Save the driver by typing
BSAVE CEN 730, A$3B0, L$1E

USING THE PRINTER:

The first time you want to use the printer you must load the driver and initialize the interface. From command mode type
BLOAD CEN 730
CALL 944

This may be done from a program by entering
100 PRINT D$;"BLOAD CEN 730" : CALL 944
assuming that D$ is a control-D.

If you want to switch back to the video monitor for output type
PR#0
or in a program enter
200 PRINT D$;"PR#0"

Then to reconnect the printer, all that is required is
CALL 954 or from a program
300 CALL 954

SETTING THE PRINT MODES:

To set normal print mode POKE 973,0
To set expanded print mode POKE 973,128

```
(************************************)
(*                                  *)
(*  LINEFEED                        *)
(*                                  *)
(*  INITIALIZE BIOS TO FILTER OUT   *)
(*  ANY LINEFEEDS SENT TO PRINTER.  *)
(*                                  *)
(*  UCSD SYSTEM OF 23 FEB 79 FOR    *)
(*  APPLE II HAS A LINE-FEED FLAG   *)
(*  AT LOCATION BF0F HEX.           *)
(*  IF THIS FLAG IS SET TO 255,     *)
(*  LINE-FEEDS WILL BE SUPPRESSED.  *)
(*  IF IT IS SET TO 0 (DEFAULT),    *)
(*  THEN LINE-FEEDS WILL BE PASSED. *)
(*                                  *)
(************************************)

PROGRAM LINEFEED;

TYPE PA=PACKED ARRAY[0..1] OF 0..255;
     TWOFACE=RECORD CASE BOOLEAN OF
             TRUE:  (INT:INTEGER);
             FALSE: (PTR:^PA);
             END;

VAR CHEAT:TWOFACE;

BEGIN
  CHEAT.INT:=-16625;  (* BF0F HEX *)
  CHEAT.PTR^[0]:=255; (* SET FLAG *)
END.
```

THE PRELIMINARY APPLE PASCAL GUIDE TO INTERFACING FOREIGN HARDWARE
10 DEC 1979 (minor revision)

This document is intended to direct users of the APPLE II who are interfacing to their machine hardware other than Apple peripheral cards. Its primary target is the community of peripheral card manufacturers who have developed a product for the Apple II under BASIC, and wish to develop an end-user Pascal interface. It is assumed that the reader is familiar with Apple II hardware and is an experienced programmer at the machine level, and has programmed in Pascal.

How Apple Pascal looks at the outside world

Currently, Apple Pascal is capable of recognizing the presence of an Apple (brand) peripheral card in slots 1 through 7, which it does at boot time by scanning the ROM in the slot address space. The purpose of this scan is (a) to determine if any card exists in the slot, since an open slot yields a random response; and (b) to find out what kind of Apple card, if any, is there, since the four types of Apple cards (printer, com, disk, and serial) have distinct values at byte locations Cn05 and Cn07. A foreign (other manufacturer's) card generally fails this test and so is disqualified in the Pascal system's list of active slots. Let us say, for example, that the Apple user has installed a foreign printer card in slot 1, and attempts to do a WRITELN to the printer. Pascal formats the request and sends it to the interpreter, which in turn reformats it and sends it to the low-level I/O package, which in Apple Pascal is called the BIOS for Basic I/O Subsystem. At the BIOS level, the slot list is checked to see what kind of card is in slot 1 (the only legal slot for the printer). It is at this point that the output request fails, since slot 1 is marked invalid unless an Apple printer, com, or serial card is in place. It is important to note in what follows that the BIOS driver routine is the only routine in the system that accesses the slot.

The BIOS

The term BIOS not only refers to the set of I/O drivers installed in the file SYSTEM.APPLE in the Pascal system, but also to the written specification for these drivers. This precise specification of the input and output to Pascal was created by the University of California, San Diego (UCSD) to make it easy to interface UCSD Pascal (parent of Apple Pascal) to any number of machines and machine configurations. The type of things specified in the BIOS includes data and control parameters, calling sequence, unit numbers, and error handling requirements.

For example, an Apple-specific version of the BIOS for the printer contains the following information:

(1) Transfer to the printer device is one character at a time.

(2) The following special characters must be recognized:
CR (carriage return) (hex 0D) Print the line and return the carriage to the first column. An automatic line feed should NOT be done.
LF (line feed) (hex 0A) Sent only after a CR. Should do a simple line feed (no return) if possible, else a CRLF.
FF (form feed) (hex 0C) Advance the paper to top-of-form (if possible) and perform a carriage return.

(3) There are only two calls to the Printer BIOS, a write and an initialization call. The initialization call should perform carriage return-line feed IF desired, but should not do a form-feed. The printer buffer should be flushed.

Interface Routines	Parameters
Printer write	data byte in register A return completion code in register X (zero if no error)
Printer init	completion code in register X (zero if online, nine if offline)

The completion code is identical to the Pascal IORESULT.

(End of Printer BIOS)

A complete description of the BIOS specification is beyond the scope of this document. The BIOS specification is available at cost from the UCSD Pascal distributor (Softech Microsystems of San Diego).

Changing the BIOS

At Apple, a BIOS has been written according to the UCSD specification to interface with all Apple peripherals. It is possible to modify the BIOS module within SYSTEM.APPLE to accommodate foreign types of peripheral cards, or to alter the behavior of an existing device such as the screen. This document contains the information necessary to enable one to manipulate the BIOS in the Pascal system in order to add non- Apple devices.

Pascal units for I/O drivers

For many types of additional hardware, such as an external clock, it is better not to alter the BIOS but to construct a Pascal unit containing routines to interface with the hardware. This unit (which may consist of assembly language routines) accesses the Cxxx addresses directly to initialize, command, and retrieve data from the hardware. The Pascal user then simply "uses" the unit and has available the routines he needs, without a system reconfiguration. (Note: The Cxxx address space may be accessed directly from Pascal with a "trix" record:

```
DEVICEBYTES = PACKED ARRAY [0..n] OF CHAR;
TRIX = RECORD CASE BOOLEAN OF
          FALSE:(ADDR: INTEGER);
          TRUE :(CONTROLREC: ^ DEVICEBYTES)
```

```
            END;
        VAR CARD: TRIX;
```

One then accesses data by the sequence

```
        CARD.ADDR := (address of hardware);
        CARD.CONTROLREC^[0] := CHR(INITMASK)      etc.
```

The type CHAR is used to ensure that only the desired byte is accessed (not the whole word).)

Patching the BIOS.

Some applications such as printers really are required to be built into the system (i.e. so that WRITELN statements access the proper output device). APPLE PASCAL is aware of the following I/O devices: CRT screen, keyboard, printer, graphics output, remote in, remote out, and block devices (disks). Remote input and remote output are general ports for serial or parallel communication. Any device that fits into one of these categories can be handled by APPLE PASCAL, by inserting a customized I/O driver into the BIOS. The procedure to do so consists of the following steps:

1. Write the I/O driver according to the BIOS specification.
2. Patch the I/O driver into SYSTEM.APPLE at a specified location (see below).
3. Patch the appropriate BIOS vector in SYSTEM.APPLE.

The "specified locations" in step 2 are in two parts. First, one may make use of the space used by the existing driver if there is one. These spaces are listed in Table I below. Generally, it is preferable to use this space in order to avoid wasting it.

If the space shown in Table I is insufficient, a small amount of unused space exists at locations hex DABE - DB7F. This is block 5, byte 190 to block 5, byte 383 (decimal), inclusive.

The BIOS vectors to patch to point to the new routine are given by Table II. Note that the method of routine entry is via a JSR, with the return address on the stack.

The actual patching of SYSTEM.APPLE must be done by a user-supplied patcher; an easy method is to use BLOCKREAD to read in the SYSTEM.APPLE file and the new driver object file, then MOVELEFT to copy the new driver in and BLOCKWRITE to write the new interpreter file. A supplier should provide to the user, as a package, such a program along with his hardware and altered BIOS routine, making it easy for an Apple user to add several modifications to his single Apple Pascal system.

Folding of the Language card memory is taken care of automatically by the interpreter.

When writing BIOS routines, there is one additional consideration: the keyboard input to Apple Pascal is done by polling, not by interrupts; therefore, at convenient locations throughout the BIOS there exist calls to the character-available check routine located at location D681 in the interpreter. When replacing the BIOS routines with customized counterparts, those substitute routines should each contain a call to the check routine.

TABLE I BIOS DRIVER AREAS IN FILE SYSTEM.APPLE

AREA	RAM LOCATION LO-ADDR	RAM LOCATION HI-ADDR	SYSTEM.APPLE FILE LO-BLK,BYTE	SYSTEM.APPLE FILE HI-BLK,BYTE
CONSOLE INIT & READ	D681	D787	3,129	3,391
CONSOLE WRITE	D7D0	D7F6	3,464	3,502
SCREEN DRIVER	D87B	DABD	4,123	5,189
PRINTER INIT	D788	D790	3,392	3,400
PRINTER WRITE	D830	D84D	4,48	4,77
REMOTE INIT	D79C	D7A2	3,412	3,418
REMOTE READ	D84E	D87A	4,78	4,122
REMOTE WRITE	D809	D810	4,9	4,16
COMMON ROUTINES FOR	D791	D79B	3,401	3,411
PRINTER & REMOTE INIT	D7A3	D7CF	3,419	3,463
SERIAL CARD WRITE	D7F7	D808	3,503	4,8
PRINTER CARD WRITE	D811	D81E	4,17	4,30
COM CARD WRITE	D81F	D82F	4,31	4,47
DISK ROUTINES	D000	D569	0,0	2,361
GRAPHIC WRITE	none			

Notes on Table I.

RAM addresses are given in hex. Locations in the file SYSTEM.APPLE are given in decimal as a block & byte offset from the beginning of the file (starting at zero).

Console init routines contain routines used by various other parts of the system, including console write. It is best not to destroy these. Replacement of the Apple keyboard will entail replacement of the console character available check routine mentioned above, with a hook at the original location. Note that the console is highly reconfigurable from the PASCAL level (program SETUP); thus, one should be very thoughtful before putting out the effort to change the console BIOS.

Console write calls the screen driver, unless an external terminal is attached.

Both the printer and remote init routines use both of the common routines listed.

Printer write uses printer card write, serial card write, and com card write. Remote write uses serial card write, com card write, and printer write (sorry about the nonmodularity).

The console and disk drivers are considered replaceable and not extendable. Therefore, the (large) space allotted in Table I is considered sufficient for console and disk needs. Users wishing to interface with the existing console or disk drivers should do so with the cooperation of Apple Computer.

TABLE III BIOS VECTORS

VECTOR	RAM LOCATION	SYSTEM.APPLE FILE BLK,BYTE
CONSOLE READ	FF7B	23,379
CONSOLE WRITE	FF84	23,388
CONSOLE INIT	FF8D	23,397

PRINTER WRITE	FF96	23,406
PRINTER INIT	FF9F	23,415
DISK WRITE	FFA8	23,424
DISK READ	FFB1	23,433
DISK INIT	FFBA	23,442
REMOTE READ	FFC3	23,451
REMOTE WRITE	FFCC	23,460
REMOTE INIT	FFD5	23,469
GRAPHICS WRITE	FFDE	23,478

All vectors are the 16-bit argument of a JSR instruction. The address given is the lo-byte address; the high byte goes in the (given address) + 1.

Zero Page Usage

If zero page storage is needed, the following information is useful Pascal makes use of variables stored at $50 thru $FF. The disk routines use variables at $36-$4F. Locations $00-$35 are considered to be pure temporaries, i.e. information may be stored in these locations but may be destroyed by other routines. Also, locations $200-$399 form a temporary area used by the disk and screen-scrolling routines, and may be used as a temporary buffer space for any other routine.

It should be noted that the drawback to all this patching is that it is dependent on the current system version. It is Apple's intention, however, to provide a standard, generalized method for system reconfiguration before the next system release occurs. Furthermore, Apple will attempt to keep persons and institutions who receive this document informed on changes and proposed changes to Apple Pascal. Conversely, if you haven't received this document from Apple Computer, you may not be kept properly informed of system changes which affect you.

BIOS listing

Attached is a listing of the Apple Pascal BIOS, which provides the exact specification of the inputs and outputs of the drivers.

```
0000:                                   .PROC BIOS
Current memory available:     9660
0000:                          ;---------------------------------------
0000:                          ;
0000:                          ;  APPLE BIOS FOR UCSD PASCAL
0000:                          ;
0000:                          ; COPYRIGHT 1978 APPLE COMPUTER INC
0000:                          ; WRITTEN BY BILL ATKINSON
0000:                          ;
0000:                          ;---------------------------------------
0000:                          ;
0000:                          ; ZERO PAGE PURE TEMPS
0000:                          ;
0000:                          ;---------------------------------------
0000: 0000                     ZEROL       .EQU 0              ;USED TO CLEAR MEM AT RESET
0000: 0001                     ZEROH       .EQU 1
0000: 0002                     JUMP1       .EQU 2              ;USED FOR CTRL CHAR CASE JUMP
0000: 0003                     JUMP2       .EQU 3
0000: 0004                     BXS1L       .EQU 4              ;EXTRA COPY POINTERS
0000: 0005                     BXS1H       .EQU 5
0000: 0006                     BXS2L       .EQU 6
0000: 0007                     BXS2H       .EQU 7
0000: 0008                     CKPTRL      .EQU 8
0000: 0009                     CKPTRH      .EQU 9
0000: 000A                     CHECKL      .EQU 10.
0000: 000B                     CHECKH      .EQU 11.
0000: 000C                     TT1         .EQU 12.
0000: 000D                     TT2         .EQU 13.
0000: 000E                     TT3         .EQU 14.
0000:
0000:                          ;---------------------------------------
0000:                          ;
0000:                          ; ZERO PAGE PERMANENTS
0000:                          ;
0000:                          ;---------------------------------------
0000: 00F0                     FIRST       .EQU 0F0            ;START ZERO PAGE USE
0000: 00F0                     BAS1L       .EQU FIRST          ;SCREEN 1 PTR
0000: 00F1                     BAS1H       .EQU FIRST+1
0000: 00F2                     BAS2L       .EQU FIRST+2        ;SCREEN 2 PTR
0000: 00F3                     BAS2H       .EQU FIRST+3
0000: 00F4                     CH          .EQU FIRST+4        ;HORIZ CURSOR, 0..79
0000: 00F5                     CV          .EQU FIRST+5        ;VERT CURSOR, 0..23
0000: 00F6                     TEMP1       .EQU FIRST+6
0000: 00F7                     TEMP2       .EQU FIRST+7
0000: 00F8                     SYSCOM      .EQU FIRST+8        ;2 BYTES PTR TO SYSCOM AREA
0000:
0000:                          ;---------------------------------------
0000:                          ;
0000:                          ; BF00 PAGE PERMANENTS
0000:                          ;
0000:                          ;---------------------------------------
0000: BF0E                     SCRMODE     .EQU 0BF0E
0000: BF0F                     LFFLAG      .EQU 0BF0F
0000: BF11                     NLEFT       .EQU 0BF11
0000: BF12                     ESCNT       .EQU 0BF12
```

```
0000: BF13                  RANDL       .EQU 0BF13
0000: BF14                  RANDH       .EQU 0BF14
0000: BF15                  CONFLGS     .EQU 0BF15
0000: BF16                  BREAK       .EQU 0BF16          ;2 BYTES
0000: BF18                  RPTR        .EQU 0BF18          ;1 BYTE
0000: BF19                  WPTR        .EQU 0BF19          ;1 BYTE
0000: BF1A                  RETL        .EQU 0BF1A
0000: BF1B                  RETH        .EQU 0BF1B
0000:
0000:                       ;---------------------------------------
0000:                       ;
0000:                       ; MISCELANEOUS PROGRAM EQUATES
0000:                       ;
0000:                       ;---------------------------------------
0000: 0200                  BUFFER      .EQU 0200           ;TEMP HSHIFT BUFFER (OVERLAPS DISK BUF)
0000: 03B1                  CONBUF      .EQU 03B1           ;78 CHAR TYPE-AHEAD BUF
0000: 004E                  CBUFLEN     .EQU 04E            ;78 DECIMAL
0000: D03C                  DWRITE      .EQU 0D03C
0000: D040                  DREAD       .EQU 0D040
0000: D004                  DINIT       .EQU 0D004
0000: D017                  DRESET      .EQU 0D017          ;POWERUP DISK INIT
0000: D152                  GOFORIT     .EQU 0D152          ;PASCAL STARTUP POINT
0000: 000C                  NCTRLS      .EQU 12.            ;# CTRL CHARS IN TABLE
0000: BFF8                  SLTTYPS     .EQU 0BFF8
0000:
0000:
0000:
0000:                       ;-----------------------------
0000:                       ;
0000:                       ; DISK ROUTINES ARE AT D000 TO D5DB
0000:                       ; PUT STARTUP STUFF AFTER THEM
0000:                       ;
0000:                       ;-----------------------------
0000:                               .ORG 0D5DC
D5DC:
D5DC:                       ;---------------------------------------
D5DC:                       ;
D5DC:                       ; HARD RESET INITIALIZATION
D5DC:                       ;
D5DC:                       ;---------------------------------------
D5DC: D8                    RESET       CLD                 ;SET HEX MODE
D5DD:
D5DD:                       ;---------------------------------------
D5DD:                       ;
D5DD:                       ; CLEAR ALL MEMORY 0 TO BFFF
D5DD:                       ;
D5DD:                       ;---------------------------------------
D5DD: A9 00                 START       LDA #0
D5DF: 85 00                             STA ZEROL
D5E1: 85 01                             STA ZEROH
D5E3: A8                                TAY
D5E4: AA                                TAX
D5E5: 91 00                 ZERLP       STA (ZEROL),Y       ;WRITE A BYTE OF 0
D5E7: C8                                INY                 ;BUMP POINTER
D5E8: D0FB                              BNE ZERLP           ;LOOP TILL NEXT PAGE
```

```
D5EA: E6 01                         INC ZEROH           ;BUMP MSB POINTER
D5EC: E8                            INX
D5ED: E0 C0                         CPX #0C0            ;DONE CLEARING MEM?
D5EF: D0F4                          BNE ZERLP
D5F1:
D5F1:                       ;-----------------------
D5F1:                       ;
D5F1:                       ; CHECKSUM PROMS ON EACH SLOT
D5F1:                       ; TO FIND OUT WHO'S OUT THERE
D5F1:                       ;
D5F1:                       ; SUM TWICE TO TELL IF CARD THERE
D5F1:                       ; IF SUMS DONT MATCH THEN NO PROM IS THERE
D5F1:                       ; IF MS BYTE OF SUM=0 THEN NO PROM IS PRESENT
D5F1:                       ;
D5F1:                       ;-----------------------
D5F1: A0 C7                         LDY #0C7            ;POINT TO SLOT 7 PROM
D5F3: 84 09                 NXTCRD  STY CKPTRH          ;(CKPTRL=0 FROM MEM CLEAR)
D5F5: 20 ****                       JSR CKPAGE          ;16 BIT SUM IN X,A
D5F8: 85 0A                         STA CHECKL
D5FA: 86 0B                         STX CHECKH          ;SAVE FOR MATCH
D5FC: 20 ****                       JSR CKPAGE          ;SUM AGAIN
D5FF: E0 00                         CPX #0              ;WAS MSB ZERO?
D601: F0**                          BEQ NOPROM          ;YES NO PROM ON CARD
D603: C5 0A                         CMP CHECKL          ;LSB MATCH?
D605: D0**                          BNE NOPROM          ;NO, NO PROM ON CARD
D607: E4 0B                         CPX CHECKH
D609: D0**                          BNE NOPROM          ;MSB DIDNT MATCH
D60B: F0**                          BEQ SKIPIORTS       ;ALWAYS TAKEN
D60D:
D60D:                       ;---------------------------
D60D:                       ;
D60D:                       ; TABLE OF CN05 AND CN07 BYTES OF EACH CARD
D60D:                       ;
D60D:                       ;---------------------------
D60D: 03 18 38 48           CN05BYTS    .BYTE 003,018,038,048
D611: 3C 38 18 48           CN07BYTS    .BYTE 03C,038,018,048
D615:
D615:                       ;---------------------------
D615:                       ;
D615:                       ; NOW THAT WE KNOW A CARD IS THERE,
D615:                       ; EXAMINE CN05 AND CN07 BYTE TO
D615:                       ; DETERMINE WHICH CARD IT IS
D615:                       ;
D615:                       ; SET CARDTYPE AS FOLLOWS:
D615:                       ; 0=CKSUM NOT REPEATABLE OR MSB=0
D615:                       ; 1=CKSUM REPEATABLE, CARD NOT RECOGNIZED
D615:                       ; 2=DISK CARD (BYTE 07= 03C)
D615:                       ; 3=COM CARD  (BYTE 07= 038)
D615:                       ; 4=SERIAL    (BYTE 07= 018)
D615:                       ; 5=PRINTER   (BYTE 07= 048)
D615:                       ;
D615:                       ;-----------------------------
D60B* 08
D615: A2 05                 SKIPIORTS   LDX #5              ;4 TYPES OF CARDS
D617: A0 05                 NXTYP       LDY #5              ;CHECK BYTE CN05 OF CARD
```

```
D619: B1 08                         LDA (CKPTRL),Y
D61B: DD 0BD6                       CMP CN05BYTS-2,X      ;MATCH TABLE?
D61E: D0**                          BNE TRYNXT            ;NO, TRY NEXT IN LIST
D620: A0 07                         LDY #7
D622: B1 08                         LDA (CKPTRL),Y        ;TEST CN07 BYTE
D624: DD 0FD6                       CMP CN07BYTS-2,X      ;MATCH TABLE?
D627: F0**                          BEQ STOR              ;BOTH MATCHED, CARD RECOGNIZED
D61E* 09
D629: CA                    TRYNXT  DEX                   ;BUMP TO NEXT IN LIST
D62A: E0 02                         CPX #2                ;TRY ALL TYPES IN LIST
D62C: B0E9                          BCS NXTYP             ;IF NOT IN LIST,FALL THRU WITH X=1
D627* 05
D62E: A4 09                 STOR    LDY CKPTRH
D630: 8A                            TXA
D631: 99 38BF                       STA SLTTYPS-0C0,Y
D609* 29
D605* 2D
D601* 31
D634: A4 09                 NOPROM  LDY CKPTRH
D636: 88                            DEY                   ;BUMP TO NEXT LOWER SLOT
D637: C0 C0                         CPY #0C0              ;SLOTS 7 DOWNTO 1 DONE?
D639: D0B8                          BNE NXTCRD            ;LOOP TILL 7 SLOTS DONE
D63B:
D63B:                       ;---------------------------------------
D63B:                       ;
D63B:                       ; SET SCREEN MODE ETC
D63B:                       ;
D63B:                       ;---------------------------------------
D63B: AD 51C0                       LDA 0C051             ;SET TEXT MODE
D63E: AD 52C0                       LDA 0C052             ;SET BOTTOM 4 GRAFIX
D641: AD 54C0                       LDA 0C054             ;SELECT PRIMARY PAGE
D644: AD 57C0                       LDA 0C057             ;SELECT HIRES GRAFIX
D647: AD 10C0                       LDA 0C010             ;CLEAR KEYBOARD STROBE
D64A: 20 ****                       JSR FORM              ;ERASE SCREEN
D64D: 20 ****                       JSR INVERT            ;PUT CURSOR ON SCREEN
D650: 20 17D0                       JSR DRESET            ;DO ONCE ONLY DISK INIT
D653: AD FBBF                       LDA SLTTYPS+3         ;WHAT CARD IN SLOT 3?
D656: A0 30                         LDY #030              ;SLOT 3
D658: 20 ****                       JSR GENIT             ;SET BAUD IF COM OR SER THERE
D65B: E0 00                         CPX #0                ;WAS AN EXTERNAL CONSOLE THERE?
D65D: D0**                          BNE STARTUP           ;NO,USE APPLE SCREEN
D65F: A9 04                         LDA #4
D661: 8D 0EBF                       STA SCRMODE           ;SET BIT 2 FOR EXT CON
D65D* 05
D664: 4C ****               STARTUP JMP JPASCAL           ;FOLD IN INTERP AND START PASCAL
D667:
D667:                       ;-------------------------
D667:                       ;
D667:                       ; SUB TO CHECKSUM ONE PAGE
D667:                       ;
D5FD* 67D6
D5F6* 67D6
D667: A9 00                 CKPAGE  LDA #0
D669: AA                            TAX                   ;CLEAR SUM
D66A: A8                            TAY                   ;CLEAR INDEX
```

```
D66B: 18                    CKNX        CLC
D66C: 71 08                             ADC (CKPTRL),Y      ;ADD BYTE
D66E: 90**                              BCC NOCRY
D670: E8                                INX                 ;INC HI BYTE IF CARRY
D66E* 01
D671: C8                    NOCRY       INY                 ;BUMP INDEX
D672: D0F7                              BNE CKNX            ;SUM 256 BYTES
D674: 60                                RTS                 ;RETURN SUM IN X,A AND Y=0
D675:
D675:
D675:
D675:                               .INCLUDE BIOS:DEVICES.TEXT
D675:
D675:
D675:                       ;---------------------------- ------ --------------------------
D675:                       ;
D675:                       ;  BIOS HANDLERS FOR LOGICAL AND PHYSICAL DEVICES.
D675:                       ;
D675:                       ;-------------------------------------------------------------
D675:
D675:
D675:
D675:                       ;---------------------------------------
D675:                       ;
D675:                       ; OLD MONITOR WAIT ROUTINE
D675:                       ; (LOCATION CHANGED)
D675:                       ;
D675:                       ;---------------------------------------
D675: 38                    WAIT        SEC
D676: 48                    WAIT2       PHA
D677: E9 01                 WAIT3       SBC #1
D679: D0FC                              BNE WAIT3           ;1.0204 USEC
D67B: 68                                PLA                 ;(13+2712;A+512;A;A)
D67C: E9 01                             SBC #1
D67E: D0F6                              BNE WAIT2
D680: 60                                RTS
D681:
D681:                       ;---------------------------------------
D681:                       ;
D681:                       ; CONSOLE CHECK FOR CHAR AVAIL
D681:                       ; STATUS AND ALL REGS PRESERVED
D681:                       ; IF CHAR AVAIL,PUT IN CONBUF AND INC WPTR.
D681:                       ;
D681:                       ; WARNING...THIS ROUTINE ALSO CALLED FROM DISK ROUTINES
D681:                       ;
D681:                       ;---------------------------------------
D681: 08                    CONCK       PHP
D682: 48                                PHA
D683: 8A                                TXA
D684: 48                                PHA
D685: 98                                TYA
D686: 48                                PHA
D687: EE 13BF               RNDINC      INC RANDL           ;BUMP 16 BIT RANDOM SEED
D68A: D0**                              BNE RNDOK
D68C: EE 14BF                           INC RANDH
D68A* 03
```

```
D68F: AD FBBF           RNDOK    LDA SLTTYPS+3      ;WHAT CARD IS IN SLOT 3?
D692: C9 03                      CMP #3             ;IS IT A COM CARD?
D694: F0**                       BEQ COMCK          ;YES, GO CHECK IT
D696: C9 04                      CMP #4             ;IS IT A SERIAL CARD?
D698: F0**                       BEQ JDONCK         ;YES, IT CANT BE TESTED
D69A: AD 00C0           TSTKBD   LDA 0C000          ;TEST APPLE KEYBOARD
D69D: 10**                       BPL JDONCK         ;NO CHAR AVAIL
D69F: 8D 10C0                    STA 0C010          ;CLEAR KEYBD STROBE
D6A2: 29 7F                      AND #07F           ;MASK OFF TOP BIT
D6A4: C9 0B                      CMP #11.           ;CTRL-K?
D6A6: D0**                       BNE NOTK
D6A8: A9 5B                      LDA #05B           ;YES, REPLACE WITH LEFT SQR BRACKETT
D6A6* 02
D6AA: C9 01             NOTK     CMP #1             ;CTRL-A?
D6AC: D0**                       BNE NTTAB
D6AE: 20 ****                    JSR HTAB           ;YES, TAB NEXT MULT 40
D6B1: AD 15BF                    LDA CONFLGS
D6B4: 29 FE                      AND #0FE
D6B6: 8D 15BF                    STA CONFLGS        ;CLEAR AUTO-FOLLOW BIT
D6B9: 4C ****                    JMP DONECK
D6AC* 0E
D6BC: C9 1A             NTTAB    CMP #26.           ;CTRL-Z?
D6BE: D0**                       BNE NOTFOL         ;NO, PUT CHAR IN BUFFER
D6C0: AD 15BF                    LDA CONFLGS
D6C3: 09 01                      ORA #1
D6C5: 8D 15BF                    STA CONFLGS        ;SET AUTO-FOLLOW BIT
D6C8: D0**                       BNE DONECK         ;BR ALWAYS
D694* 34
D6CA: AD BEC0           COMCK    LDA 0C0BE          ;CHAR AVAIL?
D6CD: 4A                         LSR A
D6CE: 90**                       BCC DONECK         ;NO CHAR AVAIL
D6D0: AD BFC0                    LDA 0C0BF          ;GET CHAR FROM UART
D6D3: 29 7F                      AND #07F           ;MASK OFF BIT 7
D6BE* 15
D6D5: A0 55             NOTFOL   LDY #055
D6D7: D1 F8                      CMP (SYSCOM),Y     ;STOP CHAR?
D6D9: D0**                       BNE NOTSTOP
D6DB: AD 15BF                    LDA CONFLGS
D6DE: 49 80                      EOR #080
D6E0: 8D 15BF                    STA CONFLGS        ;YES, TOGGLE STOP BIT (BIT 7)
D69D* 44
D698* 49
D6E3: 4C ****           JDONCK   JMP DONECK
D6D9* 0B
D6E6: 88                NOTSTOP  DEY
D6E7: D1 F8                      CMP (SYSCOM),Y
D6E9: D0**                       BNE NOTBRK
D6EB: AD 15BF                    LDA CONFLGS
D6EE: 29 3F                      AND #03F
D6F0: 8D 15BF                    STA CONFLGS        ;CLEAR FLUSH&STOP BITS
D6F3: 6C 16BF                    JMP @BREAK         ;BREAK OUT
D6E9* 0B
D6F6: 88                NOTBRK   DEY
D6F7: D1 F8                      CMP (SYSCOM),Y     ;FLUSH?
```

```
D6F9: D0**                          BNE NOTFLUS
D6FB: AD 15BF                       LDA CONFLGS
D6FE: 49 40                         EOR #040
D700: 8D 15BF                       STA CONFLGS         ;TOGGLE FLUSH BIT (BIT 6)
D703: 4C ****                       JMP DONECK
D6F9* 0B
D706: AE 19BF               NOTFLUS LDX WPTR
D709: 20 ****                       JSR BUMP
D70C: EC 18BF                       CPX RPTR            ;BUFFER FULL?
D70F: D0**                          BNE BUFOK
D711: 20 ****                       JSR BELL
D714: 4C ****                       JMP DONECK          ;BEEP&IGNORE CHAR
D70F* 06
D717: 8E 19BF               BUFOK   STX WPTR
D71A: 9D B103                       STA CONBUF,X        ;PUT CHAR IN BUFFER
D715* 1DD7
D704* 1DD7
D6E4* 1DD7
D6CE* 4D
D6C8* 53
D6BA* 1DD7
D71D: 2C 15BF               DONECK  BIT CONFLGS         ;IS STOP FLAG SET?
D720: 10**                          BPL CKEXIT
D722: 4C 87D6                       JMP RNDINC          ;LOOP IF IN STOP MODE
D720* 03
D725: 68                    CKEXIT  PLA
D726: A8                            TAY
D727: 68                            PLA
D728: AA                            TAX
D729: 68                            PLA
D72A: 28                            PLP
D72B: 60                            RTS                 ;ELSE RESTORE STAT AND ALL REG AND RETURN
D70A* 2CD7
D72C: E8                    BUMP    INX                 ;BUMP BUFFER POINTER WITH WRAP-AROUND
D72D: E0 4E                         CPX #CBUFLEN
D72F: D0**                          BNE BMPRTS
D731: A2 00                         LDX #0
D72F* 02
D733: 60                    BMPRTS  RTS
D734:
D734:
D734:
D734:
D734:
D734:
D734:
D734:                       ;-----------------------------------
D734:                       ;
D734:                       ; INITIALIZE CONSOLE:
D734:                       ;
D734:                       ;-----------------------------------
D734: 68                    CINIT   PLA
D735: 85 F6                         STA TEMP1           ;SAVE RETURN ADDR
D737: 68                            PLA
```

```
D738: 85 F7                         STA TEMP2
D73A: 68                            PLA
D73B: 85 F8                         STA SYSCOM          ;SAVE PTR TO SYSCOM AREA
D73D: 68                            PLA
D73E: 85 F9                         STA SYSCOM+1
D740: 68                            PLA
D741: 8D 16BF                       STA BREAK           ;SAVE BREAK ADDRESS
D744: 68                            PLA
D745: 8D 17BF                       STA BREAK+1
D748: A5 F7                         LDA TEMP2
D74A: 48                            PHA                 ;RESTORE RETURN ADDR
D74B: A5 F6                         LDA TEMP1
D74D: 48                            PHA
D74E: AD 18BF                       LDA RPTR            ;FLUSH TYPE-AHEAD BUFFER
D751: 8D 19BF                       STA WPTR
D754: AD 15BF                       LDA CONFLGS
D757: 29 3E                         AND #03E
D759: 8D 15BF                       STA CONFLGS         ;CLEAR STOP,FLUSH,AUTO-FOLLOW BITS
D75C: 20 ****                       JSR TAB3            ;NO,HORIZ SHIFT FULL LEFT
D75F: A2 00               CINIT2    LDX #0              ;CLEAR IORESULT
D761: 60                            RTS                 ;AND RETURN
D762:
D762:                     ;---------------------------------
D762:                     ;
D762:                     ; READ FROM CONSOLE:
D762:                     ; KEYBOARD,COM OR SERIAL CARD IN SLOT 3
D762:                     ;
D762:                     ;---------------------------------
D762: 20 ****             CREAD     JSR ADJUST          ;HORIZ SCROLL IF NECESSARY
D765: A0 30                         LDY #030            ;SLOT 3
D767: AD FBBF                       LDA SLTTYPS+3       ;WHAT TYPE OF CARD?
D76A: C9 04                         CMP #4              ;IS IT A SERIAL CARD?
D76C: D0**                          BNE CREAD2          ;NO,CONTINUE
D76E: 4C ****                       JMP RSER            ;YES, READ IT
D76C* 03
D771: 20 81D6             CREAD2    JSR CONCK           ;TEST CHAR
D774: AE 18BF                       LDX RPTR
D777: EC 19BF                       CPX WPTR
D77A: F0E6                          BEQ CREAD           ;LOOP TILL SOMETHING IN BUFFER
D77C: 20 2CD7                       JSR BUMP
D77F: 8E 18BF                       STX RPTR            ;BUMP READ POINTER
D782: BD B103                       LDA CONBUF,X        ;GET CHAR FROM BUFFER
D785: A2 00                         LDX #0              ;CLEAR IORESULT
D787: 60                            RTS                 ;AND RETURN TO PASCAL
D788:
D788:                     ;---------------------------------
D788:                     ;
D788:                     ; INITIALIZE PRINTER:
D788:                     ; PRINTER IS ALWAYS IN SLOT 1
D788:                     ; IT MAY BE A PRINTER,COM,OR SERIAL CARD
D788:                     ;
D788:                     ;---------------------------------
D788: A0 10               PINIT     LDY #010            ;SLOT 1 ; 010
D78A: AD F9BF                       LDA SLTTYPS+1       ;WHAT CARD IN SLOT 1?
D78D: C9 05                         CMP #5              ;PRINTER CARD?
```

```
D78F: F0**                          BEQ CLRIO1          ;YES,NO INIT NEEDED
D659* 91D7
D791: C9 04                 GENIT   CMP #4              ;SERIAL CARD?
D793: F0**                          BEQ ISER            ;YES, INIT SER CARD
D795: C9 03                         CMP #3              ;COM CARD?
D797: F0**                          BEQ ICOM            ;YES,INIT COM CARD
D799: A2 09                         LDX #9              ;NONE OF ABOVE,OFFLINE
D79B: 60                            RTS
D79C:
D79C:                       ;-----------------------------------
D79C:                       ;
D79C:                       ; INITIALIZE REMOTE:
D79C:                       ; REMOTE IS ALWAYS IN SLOT 2
D79C:                       ; IT MAY BE A COM OR SERIAL CARD
D79C:                       ;
D79C:                       ;-----------------------------------
D79C: AD FABF               RINIT   LDA SLTTYPS+2       ;WHAT CARD IN SLOT 2?
D79F: A0 20                         LDY #020
D7A1: D0EE                          BNE GENIT           ;BR ALWAYS TAKEN
D7A3:
D7A3:                       ;-----------------------------------
D7A3:                       ;
D7A3:                       ; INIT COM CARD, Y=ONO
D7A3:                       ;
D7A3:                       ;-----------------------------------
D797* 0A
D7A3: A9 03                 ICOM    LDA #3              ;MASTER INIT
D7A5: 99 8EC0                       STA 0C08E,Y         ;TO STATUS
D7A8: A9 15                         LDA #21.
D7AA: 99 8EC0                       STA 0C08E,Y         ;SET BAUD ETC
D78F* 1C
D7AD: A2 00                 CLRIO1  LDX #0              ;CLEAR IORESULT
D7AF: 60                            RTS                 ;AND RETURN
D7B0:
D7B0:                       ;-----------------------------------
D7B0:                       ;
D7B0:                       ; INIT SERIAL CARD, Y=ONO
D7B0:                       ;
D7B0:                       ;-----------------------------------
D793* 1B
D7B0: 20 ****               ISER    JSR SER1            ;ASSORTED GARBAGE
D7B3: 20 00C8                       JSR 0C800           ;SET UP SLOT DEPENDENTS
D7B6: A2 00                 CLRIO3  LDX #0              ;CLEAR IORESULT
D7B8: 60                            RTS                 ;AND RETURN
D7B9:
D7B9:                       ;-----------------------------------
D7B9:                       ;
D7B9:                       ; ASSORTED SERIAL CARD SET-UP
D7B9:                       ;
D7B9:                       ;-----------------------------------
D7B1* B9D7
D7B9: 8C F806               SER1    STY 06F8            ;STORE NO
D7BC: 98                            TYA
D7BD: 4A                            LSR A
D7BE: 4A                            LSR A
```

```
D7BF: 4A                        LSR A
D7C0: 4A                        LSR A
D7C1: 09 C0                     ORA #0C0
D7C3: AA                        TAX                     ;MAKE OCN IN X
D7C4: A9 00                     LDA #0
D7C6: 85 F6                     STA TEMP1
D7C8: 86 F7                     STX TEMP2               ;SET UP INDIRECT ADDRESS
D7CA: AD FFCF                   LDA 0CFFF               ;TURN OFF ALL C8 ROMS
D7CD: B1 F6                     LDA (TEMP1),Y           ;SELECT C8 BANK
D7CF: 60                        RTS
D7D0:
D7D0:                   ;-----------------------------------
D7D0:                   ;
D7D0:                   ; WRITE TO CONSOLE:
D7D0:                   ; VIDEO SCREEN,COM OR SER CARD IN SLOT 3
D7D0:                   ;
D7D0:                   ;-----------------------------------
D7D0: 20 B1D6           CWRITE  JSR CONCK               ;CONSOLE CHAR AVAIL?
D7D3: 2C 15DF                   BIT CONFLGS             ;IS FLUSH FLAG SET?
D7D6: 70**                      BVS CLRIO               ;YES,DISCARD CHAR & RETURN
D7D8: AA                        TAX                     ;SAVE CHAR IN X
D7D9: A0 30                     LDY #030                ;SLOT 3;010
D7DB: AD FBBF                   LDA SLTTYPS+3           ;WHAT KIND OF CARD?
D7DE: C9 03                     CMP #3                  ;COM CARD?
D7E0: F0**                      BEQ WCOM                ;YES WRITE TO COM CARD SLOT 3
D7E2: C9 04                     CMP #4                  ;SERIAL CARD?
D7E4: F0**                      BEQ WSER                ;YES,WRITE TO SER CARD SLOT 3
D7E6: 8A                        TXA                     ;ELSE RESTORE CHAR & SEND TO SCREEN
D7E7: 85 F6             VIDOUT  STA TEMP1               ;SAVE CHAR FOR LATER
D7E9: 20 ****                   JSR INVERT              ;REMOVE CURSOR
D7EC: A4 F4                     LDY CH
D7EE: 20 ****                   JSR VOUT2               ;DO THE BUSINESS
D7F1: 20 ****                   JSR INVERT              ;RESTORE THE CURSOR
D7D6* 1C
D7F4: A2 00             CLRIO   LDX #0                  ;CLR IORESULT
D7F6: 60                        RTS                     ;RETURN FROM VIDOUT
D7F7:
D7F7:                   ;-----------------------------------
D7F7:                   ;
D7F7:                   ; WRITE TO SERIAL CARD, Y=ONO,CHAR IN X
D7F7:                   ;
D7F7:                   ;-----------------------------------
D7E4* 11
D7F7: 20 81D6           WSER    JSR CONCK               ;CONSOLE CHAR?
D7FA: 8A                        TXA
D7FB: 48                        PHA                     ;SAVE CHAR ON STACK
D7FC: 20 B9D7                   JSR SER1                ;ASSORTED GARBAGE
D7FF: 68                        PLA
D800: 9D B805                   STA 05B8,X              ;SET UP DATA BYTE
D803: 20 AAC9                   JSR 0C9AA               ;SEND IT (SHOUT)
D806: A2 00                     LDX #0
D808: 60                        RTS
D809:
D809:                   ;-----------------------------------
D809:                   ;
```

```
D809:                    ; WRITE TO REMOTE:, CHAR IN A
D809:                    ;
D809:                    ;---------------------------------
D809: AA                 RWRITE    TAX                 ;SAVE CHAR
D80A: AD FABF                      LDA SLTTYPS+2       ;WHAT CARD IN SLOT 2?
D80D: A0 20                        LDY #020
D80F: D0**                         BNE GENW2           ;BR ALWAYS TAKEN
D811:
D811:                    ;---------------------------------
D811:                    ;
D811:                    ; WRITE TO PRINTER CARD SLOT1, CHAR IN X
D811:                    ;
D811:                    ;---------------------------------
D811: 20 81D6            WPRN      JSR CONCK           ;CONSOLE CHAR AVAIL?
D814: AD C1C1                      LDA 0C1C1           ;TEST PRINTER READY
D817: 30F8                         BMI WPRN            ;LOOP TILL READY
D819: 8E 90C0                      STX 0C090           ;SEND CHAR
D81C: A2 00              CLRIO2    LDX #0
D81E: 60                           RTS
D81F:
D81F:                    ;---------------------------------
D81F:                    ;
D81F:                    ; WRITE TO COM CARD, Y=ONO, CHAR IN X
D81F:                    ;
D81F:                    ;---------------------------------
D7E0* 3D
D81F: 20 81D6            WCOM      JSR CONCK           ;CONSOLE CHAR?
D822: B9 8EC0                      LDA 0C08E,Y         ;TEST UART STATUS
D825: 29 02                        AND #2              ;READY?
D827: F0F6                         BEQ WCOM            ;NO, WAIT TILL READY
D829: 8A                           TXA
D82A: 99 8FC0                      STA 0C08F,Y         ;SEND CHAR
D82D: A2 00                        LDX #0
D82F: 60                           RTS
D830:
D830:                    ;---------------------------------
D830:                    ;
D830:                    ; WRITE TO PRINTER:, CHAR IN A
D830:                    ;
D830:                    ;---------------------------------
D830: AA                 PWRITE    TAX                 ;SAVE CHAR IN X
D831: AD 0FBF                      LDA LFFLAG          ;TEST LINE-FEED FLAG
D834: 10**                         BPL LFPASS          ;PASS IF BIT7=0
D836: E0 0A                        CPX #10.            ;IS IT A LINE-FEED?
D838: F0DA                         BEQ CLRIO           ;YES, IGNORE
D834* 04
D83A: A0 10              LFPASS    LDY #010            ;SLOT 1
D83C: AD F98F                      LDA SLTTYPS+1       ;WHAT KIND OF CARD?
D83F: C9 05              GENW      CMP #5              ;PRINTER CARD?
D841: F0CE                         BEQ WPRN            ;YES WRITE TO PRINTER CARD
D80F* 32
D843: C9 04              GENW2     CMP #4              ;SERIAL CARD?
D845: F0B0                         BEQ WSER            ;YES WRITE TO SER CARD
D847: C9 03                        CMP #3              ;COM CARD?
D849: F0D4                         BEQ WCOM            ;YES WRITE TO COM CARD
```

```
D84B: A2 09              OFFLINE    LDX #9
D84D: 60                            RTS
D84E:
D84E:                    ;----------------------------------
D84E:                    ;
D84E:                    ; READ FROM REMOTE:
D84E:                    ;
D84E:                    ;----------------------------------
D84E: AD FABF            RREAD      LDA SLTTYPS+2        ;WHAT CARD IN SLOT 2?
D851: A0 20                         LDY #020
D853: C9 04              GENR       CMP #4               ;SERIAL CARD?
D855: F0**                          BEQ RSER             ;GET FROM SER CARD
D857: C9 03                         CMP #3               ;COM CARD?
D859: F0**                          BEQ RCOM             ;GET FROM COM CARD
D85B: D0EE                          BNE OFFLINE          ;CARD NOT RECOG
D85D:
D85D:                    ;----------------------------------
D85D:                    ;
D85D:                    ; READ FROM COM CARD, Y=NO
D85D:                    ;
D85D:                    ;----------------------------------
D859* 02
D85D: 20 81D6            RCOM       JSR CONCK            ;CHECK FOR CONSOLE CHAR
D860: B9 8EC0                       LDA 0C08E,Y          ;TEST UART STATUS
D863: 4A                            LSR A                ;TEST BIT 0
D864: 90F7                          BCC RCOM             ; WAIT FOR CHAR
D866: B9 8FC0                       LDA 0C08F,Y          ;GET CHAR
D869: A2 00                         LDX #0
D86B: 60                            RTS
D86C:
D86C:                    ;----------------------------------
D86C:                    ;
D86C:                    ; READ FROM SERIAL CARD, Y=ONO
D86C:                    ;
D86C:                    ;----------------------------------
D855* 15
D76F* 6CD8
D86C: 20 81D6            RSER       JSR CONCK            ;CONSOLE CHAR AVAIL?
D86F: 20 B9D7                       JSR SER1             ;ASSORTED GARBAGE
D872: 20 4DC8                       JSR 0C84D            ;GET A BYTE (SHIFTIN)
D875: BD B805                       LDA 0588,X           ;GET BYTE 0678+SLOT
D878: A2 00                         LDX #0
D87A: 60                            RTS
D87B:
D87B:
D87B:
D87B:
D87B:
D87B:                           .INCLUDE BIOS:SCREEN.TEXT
D87B:
D87B:                    ;-------------------------------------------------------
D87B:                    ;
D87B:                    ;  APPLE II SCREEN DRIVER ROUTINES FOR 80 COL DISPLAY
D87B:                    ;
```

```
D87B:                   ;---------------------------------------- ---------------
D87B:
D87B:
D87B:
D87B:                   ;--------- ----------------------------------
D87B:                   ;
D87B:                   ;    LIST OF CONTROL CHARS
D87B:                   ; ( APPLE SCREEN DRIVER EMULATES A DATAMEDIA TERMINAL )
D87B:                   ;
D87B:                   ;-------------------------------------------
D87B: 1B                CTRLCH      .BYTE 27.           ;ESCAPE
D87C: 1E                            .BYTE 30.           ;GOTOXY
D87D: 0D                            .BYTE 13.           ;CR
D87E: 0A                            .BYTE 10.           ;LF
D87F: 07                            .BYTE 7.            ;BELL
D880: 1F                            .BYTE 31.           ;REVLF
D881: 1C                            .BYTE 28.           ;NDFS
D882: 08                            .BYTE 8.            ;NDBS
D883: 0C                            .BYTE 12.           ;FORM
D884: 19                            .BYTE 25.           ;HOME
D885: 0B                            .BYTE 11.           ;CLEOS
D886: 1D                            .BYTE 29.           ;CLEOL
D887:
D887:                   ;-------------------------------------------
D887:                   ;
D887:                   ; JUMP TABLE FOR CONTROL CHARS
D887:                   ;
D887:                   ;-------------------------------------------
D887: ****              CTRLJMP     .WORD  ESCAPE
D889: ****                          .WORD  GOTOXY
D88B: ****                          .WORD  CR
D88D: ****                          .WORD  LF
D88F: ****                          .WORD  BELL
D891: ****                          .WORD  REVLF
D893: ****                          .WORD  ADVANCE
D895: ****                          .WORD  NDBS
D897: ****                          .WORD  FORM
D899: ****                          .WORD  HOME
D89B: ****                          .WORD  CLEOS
D89D: ****                          .WORD  CLEOL
D89F:
D89F:                   ;-------------------------------------------
D89F:                   ;
D89F:                   ;  START OF SCREEN HANDLING ROUTINES
D89F:                   ;
D89F:                   ;-------------------------------------------
D7EF* 9FD8
D89F: AD 128F           VOUT2       LDA ESCNT           ;STILL IN ESC SEQ?
D8A2: F0**                          BEQ NOTSEQ          ;NO, SKIP
D8A4: C9 02                         CMP #2              ;ARE WE IN XY MODE?
D8A6: 90**                          BCC SEQRET          ;NO, THERE ARE NO OTHER ESC SEQUENCES
D8A8: F0**                          BEQ SETY            ;IF ESCNT=2 THEN SET CV
D8AA: A5 F6                         LDA TEMP1           ;ELSE GET BACK CHAR
D8AC: 38                            SEC
D8AD: E9 20                         SBC #32.            ;SUB 32
```

```
D8AF: 30**                          BMI OOPSX           ;BRANCH IF NEG
D8B1: C9 50                         CMP #80.
D8B3: 90**                          BCC XOK
D8AF* 04
D8B5: A9 00             OOPSX       LDA #0              ;OUT OF RANGE, MAKE=0
D8B3* 02
D8B7: 85 F7             XOK         STA TEMP2           ;SAVE TILL GET Y TOO
D8B9: 4C ****                       JMP SEQRET          ;DEC ESCNT AND RETURN
D8A8* 12
D8BC: A5 F6             SETY        LDA TEMP1           ;GET BACK CHAR
D8BE: 38                            SEC
D8BF: E9 20                         SBC #32.
D8C1: 30**                          BMI OOPSY
D8C3: C9 18                         CMP #24.
D8C5: 90**                          BCC YOK
D8C1* 04
D8C7: A9 00             OOPSY       LDA #0
D8C5* 02
D8C9: 85 F5             YOK         STA CV              ;SET NEW CV
D8CB: 20 ****                       JSR BASCAL          ;CALC NEW PTRS
D8CE: A5 F7                         LDA TEMP2           ;GET X COORD FROM LAST
D8D0: 85 F4                         STA CH              ;TIME THRU AND SET HORIZ
D8D2: A9 00                         LDA #0
D8D4: 8D 12BF                       STA ESCNT
D8D7: 60                            RTS                 ;AND RETURN
D8BA* D8D8
D8A6* 30
D8D8: CE 12BF           SEQRET      DEC ESCNT
D8DB: 60                            RTS
D8A2* 38
D8DC: A5 F6             NOTSEQ      LDA TEMP1           ;GET CHAR BACK
D8DE: 29 7F                         AND #07F            ;MASK OFF HI BIT
D8E0: C9 20                         CMP #32.            ;CONTROL CHAR?
D8E2: 90**                          BCC CNTRL           ;YES, PROCESS IT
D8E4: C9 60                         CMP #96             ;LOWER CASE?
D8E6: 90**                          BCC UPPER
D8E8: E9 20                         SBC #32.            ;CONVERT TO UPPER
D8E6* 02
D8EA: 29 3F             UPPER       AND #03F            ;TURN OFF FLASH
D8EC: 09 80                         ORA #080            ;MAKE NOT INVERTED
D8EE: 4C ****                       JMP STOADV          ;PRINT AND RETURN
D8E2* 0D
D8F1: A2 0B             CNTRL       LDX #NCTRLS-1       ;POINT TO END OF CTRL TABLE
D8F3: DD 7BD8           CKCTRL      CMP CTRLCH,X        ;COMPARE CHAR AGAINST TABLE
D8F6: F0**                          BEQ DOCTRL          ;PROCESS IF MATCH
D8F8: CA                            DEX
D8F9: 10F8                          BPL CKCTRL          ;CHECK ALL CHARS IN TABLE
D8FB: 60                            RTS                 ;IGNORE IF NOT MATCHED
D8F6* 04
D8FC: 8A                DOCTRL      TXA
D8FD: 0A                            ASL A               ;DOUBLE INDEX FOR WORD OFFSET
D8FE: AA                            TAX
D8FF: BD 88D8                       LDA CTRLJMP+1,X     ;GET HI BYTE JMP ADDR
D902: 85 03                         STA JUMP2
D904: BD 87D8                       LDA CTRLJMP,X       ;AND LO BYTE
```

```
D907: 85 02                         STA JUMP1
D909: 6C 0200                       JMP @JUMP1          ;DO TABLE JUMP
D90C:
D90C:                       ;---------------------------------------
D90C:                       ;
D90C:                       ; ROUTINES TO PROCESS CONTROL CHARS
D90C:                       ;
D90C:                       ;---------------------------------------
D887* 0CD9
D90C: A9 01                 ESCAPE  LDA #1
D90E: 8D 12BF                       STA ESCNT           ;SET ESCAPE FLAG
D911: 60                            RTS                 ;AND RETURN
D889* 12D9
D912: A9 03                 GOTOXY  LDA #3
D914: 8D 12BF                       STA ESCNT           ;SET ESC COUNT=3 FOR XY
D917: 60                            RTS                 ;AND RETURN
D88B* 18D9
D918: A0 00                 CR      LDY #0
D91A: 84 F4                         STY CH              ;RESET HORIZ CURSOR
D91C: 60                            RTS                 ;AND RETURN
D88D* 1DD9
D91D: A5 F5                 LF      LDA CV              ;YES, GET VERT CURSOR
D91F: C9 17                         CMP #23.            ;ON BOTTOM LINE?
D921: F0**                          BEQ SCROLL          ;YES, SCROLL UP
D923: E6 F5                 LF2     INC CV              ;NO, BUMP VERT CURSOR
D925: 4C ****                       JMP BASCAL          ;CALC PTRS & RETURN
D921* 05
D928: A5 F4                 SCROLL  LDA CH
D92A: 48                            PHA                 ;SAVE HORIZ CURSOR
D92B: A2 00                         LDX #0
D92D: 86 F5                         STX CV              ;INIT LINE COUNT
D92F: 20 ****                       JSR BASCAL
D932: 20 ****               COPY1   JSR COPY            ;COPY POINTERS
D935: E8                            INX
D936: 86 F5                         STX CV              ;BUMP LINE COUNT
D938: 20 ****                       JSR BASCAL
D93B: A0 27                         LDY #39.            ;INIT CHAR COUNT
D93D: B1 F0                 COPY2   LDA (BAS1L),Y
D93F: 91 04                         STA (BXS1L),Y       ;COPY BYTE OF PG1
D941: B1 F2                         LDA (BAS2L),Y
D943: 91 06                         STA (BXS2L),Y       ;COPY BYTE OF PG2
D945: 88                            DEY
D946: 10F5                          BPL COPY2           ;LOOP 40 CHARACTERS
D948: E0 17                 SKPCOUT CPX #23.            ;LINE 23 COPIED?
D94A: D0E6                          BNE COPY1           ;NO, LOOP WHOLE PAGE
D94C: A9 00                         LDA #0
D94E: 85 F4                         STA CH              ;CLEAR HORIZ CURSOR
D950: 20 ****                       JSR CLEOL           ;CLEAR BOTTOM LINE
D953: 68                            PLA
D954: 85 F4                         STA CH              ;RESTORE HORIZ CURSOR
D956: 60                            RTS                 ;AND RETURN
D88F* 57D9
D712* 57D9
D957: A9 40                 BELL    LDA #040            ;DELAY .01 SEC
D959: 20 75D6                       JSR WAIT
```

```
D95C: A0 C0                     LDY #0C0
D95E: A9 0C             BELL2   LDA #00C        ;TOGGLE SPEAKER AT
D960: 20 75D6                   JSR WAIT        ; 1 KHZ FOR .1 SEC
D963: AD 30C0                   LDA 0C030       ;TOGGLE SPKR
D966: 88                        DEY
D967: D0F5                      BNE BELL2
D969: 60                JRET    RTS             ;AND RETURN
D891* 6AD9
D96A: A5 F5             REVLF   LDA CV          ;IS CURSOR ALREADY AT TOP?
D96C: F0FB                      BEQ JRET        ;YES, RETURN
D96E: C6 F5                     DEC CV          ;NO, BUMP UP
D970: 4C ****                   JMP BASCAL      ;CALC PTRS AND RETN
D895* 73D9
D973: C0 00             NDBS    CPY #0          ;YES, AT LEFT OF PG1?
D975: F0**                      BEQ JINV        ;YES, RETURN
D977: C6 F4                     DEC CH          ;NO, BUMP IT LEFT
D975* 02
D979: 60                JINV    RTS             ;AND RETURN
D897* 7AD9
D64B* 7AD9
D97A: 20 ****           FORM    JSR HOME
D97D: 4C ****                   JMP CLEOS
D97B* 80D9
D899* 80D9
D980: A0 00             HOME    LDY #0          ;RESET CV, CH
D982: 84 F5                     STY CV
D984: 84 F4                     STY CH
D986: 4C ****           BSRET   JMP BASCAL      ;CALC PTRS & RETN
D97E* 89D9
D89B* 89D9
D989: A5 F4             CLEOS   LDA CH
D98B: 48                        PHA             ;SAVE HORIZ CURSOR
D98C: A5 F5                     LDA CV
D98E: 48                        PHA             ;SAVE VERT CURSOR TOO
D98F: 20 ****           CLNXL   JSR CLEOL       ;CLEAR A LINE
D992: 20 18D9                   JSR CR          ;POINT TO LEFT
D995: 20 23D9                   JSR LF2         ;OF NEXT LINE DOWN
D998: A5 F5                     LDA CV
D99A: C9 18                     CMP #24.
D99C: 90F1                      BCC CLNXL       ;LOOP TILL CV>=24
D99E: 68                        PLA
D99F: 85 F5                     STA CV          ;RESTORE VERT CURSOR
D9A1: 68                        PLA
D9A2: 85 F4                     STA CH          ;RESTORE HORIZ CURSOR
D9A4: 4C ****                   JMP BASCAL      ;CALC NEW POINTERS & RTN
D990* A7D9
D951* A7D9
D89D* A7D9
D9A7: A6 F4             CLEOL   LDX CH          ;SAVE CH
D9A9: A9 A0             CLOOP   LDA #160.       ;GET A SPACE
D9AB: 20 ****                   JSR STOADV      ;PUT IT ON THE SCREEN
D9AE: 90F9                      BCC CLOOP       ;LOOP TILL END OF LINE
D9B0: 86 F4                     STX CH          ;RESTORE CH
D9B2: 60                        RTS
D9B3:
```

```
D9B3!                    ;-----------------------------------------
D9B3!                    ;
D9B3!                    ; STORE CHAR IN A ON SCREEN THEN
D9B3!                    ; FALL THROUGH TO ADVANCE
D9B3!                    ; X REG UNCHANGED, Y=CH ON EXIT
D9B3!                    ;
D9B3!                    ;-----------------------------------------
D9AC* B3D9
D8EF* B3D9
D9B3! 20 ****            STOADV      JSR STORE           ;PUT CHAR ON SCREEN
D9B6!
D9B6!                    ;-----------------------------------------
D9B6!                    ;
D9B6!                    ; ADVANCE CURSOR TO RIGHT.
D9B6!                    ; IF CH ALREADY=79 THEN DONT ADVANCE,
D9B6!                    ; AND RETURN WITH Y=79 AND CARRY SET
D9B6!                    ; ELSE RETURN WITH Y=CH, CARRY CLEAR
D9B6!                    ; X & A UNCHANGED.
D9B6!                    ;
D9B6!                    ;-----------------------------------------
D893* B6D9
D9B6! A4 F4              ADVANCE     LDY CH
D9B8! C0 4F                          CPY #79.
D9BA! B0**                           BCS ADVRTS
D9BC! C8                             INY
D9BD! 84 F4                          STY CH
D9BA* 03
D9BF! 60                 ADVRTS      RTS
D9C0!
D9C0!                    ;-----------------------------------------
D9C0!                    ;
D9C0!                    ; PUT CHAR ON SCREEN, EITHER VISABLE
D9C0!                    ; OR INVISABLE PORTION.
D9C0!                    ;
D9C0!                    ; SPECIAL CASE: IF A=0 THEN JUST INVERT
D9C0!                    ; AT THAT LOCATION INSTEAD OF STORING
D9C0!                    ;
D9C0!                    ;-----------------------------------------
D9B4* C0D9
D9C0! 48                 STORE       PHA                 ;SAVE CHAR FOR LATER
D9C1! A5 F4                          LDA CH
D9C3! 38                             SEC
D9C4! ED 11BF                        SBC NLEFT           ;WHERE IS THE HORIZ CURSOR?
D9C7! 30**                           BMI OFFLFT          ;IT'S OFF TO THE LEFT OF VISABLE SCREEN
D9C9! C9 28                          CMP #40.
D9CB! B0**                           BCS OFFRT           ;IT'S OFF TO RIGHT OF VISABLE SCREEN
D9CD! A8                             TAY                 ;ELSE IT'S ON VISABLE PORTION
D9CE! 68                             PLA                 ;GET CHAR BACK
D9CF! D0**                           BNE STORE1          ;STORE IT UNLESS 0
D9D1! B1 F0                          LDA (BAS1L),Y       ;IF 0 READ SCREEN
D9D3! 49 80                          EOR #080            ;INVERT CHARACTER
D9CF* 04
D9D5! 91 F0              STORE1      STA (BAS1L),Y       ;STOR VISABLE CHAR
D9D7! 60                             RTS
D9C7* 0F
```

```
D9D8: 18                    OFFLFT      CLC
D9D9: 69 28                             ADC #40.
D9DB: 4C ****                           JMP STOR2
D9CB* 11
D9DE: 38                    OFFRT       SEC
D9DF: E9 28                             SBC #40.
D9DC* E1D9
D9E1: A8                    STOR2       TAY
D9E2: 68                                PLA                 ;GET BACK CHAR
D9E3: D0**                              BNE STORE2          ;STORE IF NOT 0
D9E5: B1 F2                             LDA (BAS2L),Y       ;IF 0, READ SCREEN
D9E7: 49 80                             EOR #080            ;AND INVERT IT
D9E3* 04
D9E9: 91 F2                 STORE2      STA (BAS2L),Y       ;STORE HIDDEN CHAR
D9EB: 60                                RTS
D9EC:
D9EC:                       ;-----------------------------------------
D9EC:                       ;
D9EC:                       ; INVERT CHAR AT CURSOR POSITION
D9EC:                       ;
D9EC:                       ;-----------------------------------------
D7F2* ECD9
D7EA* ECD9
D9EC: A9 00                 INVERT      LDA #0              ;FLAG TO TELL STORE TO INVERT
D9EE: 4C C0D9                           JMP STORE           ;INVERT AT CH AND RETURN
D9F1:
D9F1:                       ;-----------------------------------------
D9F1:                       ;
D9F1:                       ; CALCULATE BASE ADDR POINTERS
D9F1:                       ; FOR PAGE 1 AND PAGE 2.
D9F1:                       ; ENTER WITH CV IN RANGE 0..23
D9F1:                       ; EXIT WITH BAS1L,H AND BAS2L,H SET UP
D9F1:                       ; X AND Y REGS UNCHANGED
D9F1:                       ;
D9F1:                       ;-----------------------------------------
D9A5* F1D9
D987* F1D9
D971* F1D9
D939* F1D9
D930* F1D9
D926* F1D9
D8CC* F1D9
D9F1: A5 F5                 BASCAL      LDA CV
D9F3: 4A                                LSR A               ;SET CARRY FOR 6 LINES DOWN
D9F4: 29 03                             AND #3
D9F6: 09 04                             ORA #4
D9F8: 85 F1                             STA BAS1H
D9FA: A5 F5                             LDA CV
D9FC: 29 18                             AND #018
D9FE: 90**                              BCC BSCL2
DA00: 69 7F                             ADC #07F
D9FE* 02
DA02: 85 F0                 BSCL2       STA BAS1L
DA04: 0A                                ASL A
DA05: 0A                                ASL A
```

```
DA06: 05 F0                             ORA BAS1L
DA08: 85 F0                             STA BAS1L
DA0A: 85 F2                             STA BAS2L
DA0C: A5 F1                             LDA BAS1H
DA0E: 18                                CLC
DA0F: 69 04                             ADC #4
DA11: 85 F3                             STA BAS2H           ;POINTER TO PAGE 2
DA13: 60                                RTS
DA14:
DA14:                           ;---------------------------------------
DA14:                           ;
DA14:                           ;  COPY BAS1L,H INTO BXS1L,H
DA14:                           ;  AND BAS2L,H INTO BXS2L,H
DA14:                           ;  TO MAKE AN EXTRA PAIR OF POINTERS
DA14:                           ;  FOR SCROLLING ETC.
DA14:                           ;
DA14:                           ;--------- -----------------------------
D933* 14DA
DA14: A5 F1                     COPY        LDA BAS1H           ;COPY POINTERS
DA16: 85 05                                 STA BXS1H
DA18: A5 F3                                 LDA BAS2H
DA1A: 85 07                                 STA BXS2H
DA1C: A5 F0                                 LDA BAS1L
DA1E: 85 04                                 STA BXS1L
DA20: 85 06                                 STA BXS2L           ;NOTE BAS2L ALWAYS SAME AS BAS1L
DA22: 60                        RTS7        RTS
DA23:
DA23:                           ;---------------------------------------
DA23:                           ;
DA23:                           ; HORIZONTAL SCREEN SHIFT ROUTINE
DA23:                           ;
DA23:                           ;---------------------------------------
DA23: AA                        HSHIFT      TAX                 ;SAVE HORIZ DELTA
DA24: F0FC                                  BEQ RTS7            ;RETURN IF DELTA=0
DA26: A5 F5                                 LDA CV
DA28: 48                                    PHA                 ;SAVE CV ON STACK
DA29: 8A                                    TXA
DA2A: 48                                    PHA                 ;SAVE OFFSET ON STACK
DA2B: 38                                    SEC
DA2C: E9 28                                 SBC #40.
DA2E: 10**                                  BPL OKDELT
DA30: 18                                    CLC
DA31: 69 50                                 ADC #80.            ;CALC (DELTA+40)MOD80
DA2E* 03
DA33: 48                        OKDELT      PHA                 ;SAVE ON STACK
DA34: A9 17                                 LDA #23.
DA36: 85 F5                                 STA CV              ;INIT ROW COUNTER
DA38: 20 F1D9                   LOP1        JSR BASCAL          ;CALC POINTERS
DA3B: A0 27                                 LDY #39.
DA3D: B1 F0                     LOP2        LDA (BAS1L),Y       ;GET VISABLE CHAR
DA3F: 99 0002                               STA BUFFER,Y        ;COPY INTO LEFT HALF OF BUFFER
DA42: B1 F2                                 LDA (BAS2L),Y       ;GET HIDDEN CHAR
DA44: 99 2802                               STA BUFFER+40.,Y    ;COPY INTO RIGHT HALF OF BUFFER
DA47: 88                                    DEY
DA48: 10F3                                  BPL LOP2            ,REPEAT FOR WHOLE ROW
```

```
DA4A: 68                            PLA
DA4B: 48                            PHA
DA4C: AA                            TAX                 ;GET (DELTA+40)MOD 80 IN X
DA4D: A0 27                         LDY #39.
DA4F: CA                VISLP       DEX
DA50: 10**                          BPL NOWRP1
DA52: A2 4F                         LDX #79.            ;BUFFER INDEX WRAP-AROUND
DA50* 02
DA54: BD 0002           NOWRP1      LDA BUFFER,X        ;GET FROM BUFFER
DA57: 91 F0                         STA (BAS1L),Y       ;WRITE VISABLE CHAR
DA59: 88                            DEY
DA5A: 10F3                          BPL VISLP           ;LOOP 40 VISABLE CHARS
DA5C: A0 27                         LDY #39.
DA5E: CA                HIDNLP      DEX
DA5F: 10**                          BPL NOWRP2
DA61: A2 4F                         LDX #79.            ;BUFFER INDEX WRAP-AROUND
DA5F* 02
DA63: BD 0002           NOWRP2      LDA BUFFER,X
DA66: 91 F2                         STA (BAS2L),Y       ;WRITE A HIDDEN CHAR
DA68: 88                            DEY
DA69: 10F3                          BPL HIDNLP          ;LOOP 40 HIDDEN CHARS
DA6B: C6 F5                         DEC CV
DA6D: 10C9                          BPL LOP1            ;REPEAT ALL ROWS IN PAGE
DA6F: 68                            PLA                 ;DISCARD DELTA+40 MOD 80
DA70: 68                            PLA                 ;GET PLAIN DELTA
DA71: 18                            CLC
DA72: 6D 11BF                       ADC NLEFT
DA75: 8D 11BF                       STA NLEFT           ;UPDATE COUNT OF CHARS TO LEFT
DA78: 68                            PLA
DA79: 85 F5                         STA CV              ;RESTORE CV
DA7B: 4C F1D9                       JMP BASCAL          ;AND POINTERS & RETN
DA7E:
DA7E:                       ;---------------------------------------
DA7E:                       ;
DA7E:                       ; HORIZ TAB TO NEXT MULT OF 40
DA7E:                       ;
DA7E:                       ;---------------------------------------
DA7E: AD 11BF               HTAB        LDA NLEFT
DA81: C9 14                             CMP #20.
DA83: B0**                              BCS TAB3
DA85: A9 28                 TAB2        LDA #40.
DA87: D0**                              BNE DOIT            ;ALWAYS TAKEN
DA83* 04
DA89: A9 00                 TAB3        LDA #0
DA87* 02
DA8B: 38                    DOIT        SEC
DA8C: ED 11BF                           SBC NLEFT           ;CALC DELTA
DA8F: 4C 23DA                           JMP HSHIFT          ;SHIFT AND RETURN
DA92:
DA92:                       ;---------------------------------------
DA92:                       ;
DA92:                       ; ADJUST ROUTINE DOES HORIZ
DA92:                       ; SCROLL TO KEEP CURSOR VISABLE
DA92:                       ;
DA92:                       ;---------------------------------------
```

```
DA92: AD 15BF              ADJUST    LDA CONFLGS
DA95: 4A                             LSR A              ;IS AUTO-FOLLOW (BIT 1) TRUE?
DA96: 90**                           BCC ADJRTS         ;NO, IGNORE
DA98: A5 F4                          LDA CH
DA9A: C9 14                          CMP #20.           ;CURSOR LESS THAN 20?
DA9C: B0**                           BCS ADJOK          ;NO, CONTINUE
DA9E: 4C 89DA                        JMP TAB3           ;YES TAB FULL LEFT & RTN
DA9C* 03
DAA1: 38                   ADJOK     SEC
DAA2: ED 11BF                        SBC NLEFT
DAA5: 30**                           BMI ADJLFT
DAA7: C9 25                          CMP #37.
DAA9: B0**                           BCS ADJRT
DA96* 13
DAAB: 60                   ADJRTS    RTS                ;DONT ADJUST
DAA9* 01
DAAC: A4 F4                ADJRT     LDY CH
DAAE: C0 4D                          CPY #77.
DAB0: B0**                           BCS FARRT
DAB2: 38                             SEC
DAB3: E9 24                          SBC #36.
DAB5: 4C 23DA                        JMP HSHIFT
DAB0* 06
DAB8: 4C 85DA              FARRT     JMP TAB2
DAA5* 14
DABB: 4C 23DA              ADJLFT    JMP HSHIFT
DABE:
DABE:
DABE:
DABE:
DABE:
DABE:                          .INCLUDE BIOS:TOP.TEXT
DABE:
DABE:
```

```
DABE: 00 00 00 00 00 00 00              ORG OFF00
FF00:                          ;-----------------------------------------------
FF00:                          ;
FF00:                          ;  TOP PORTION OF BIOS FOLDS IN AND OUT BIOS IN D000 BANK
FF00:                          ;
FF00:                          ;-----------------------------------------------
FF00:
FF00:
FF00:                          ;-----------------------------------------------
FF00:                          ;
FF00:                          ;  MAIN BIOS JUMP TABLE CALLED FROM INTERPRETER
FF00:                          ;
FF00:                          ;-----------------------------------------------
FF00: 4C ****                            JMP QCREAD          ;CONSOLE READ
FF03: 4C ****                            JMP QCWRITE         ;CONSOLE WRITE
FF06: 4C ****                            JMP QCINIT          ;CONSOLE INIT
FF09: 4C ****                            JMP QPWRITE         ;PRINTER WRITE
FF0C: 4C ****                            JMP QPINIT          ;PRINTER INIT
FF0F: 4C ****                            JMP QDWRITE         ;DISK WRITE
FF12: 4C ****                            JMP QDREAD          ;DISK READ
```

```
FF15: 4C ****                          JMP QDINIT          ;DISK INIT
FF18: 4C ****                          JMP QRREAD          ;REMOTE READ
FF1B: 4C ****                          JMP QRWRITE         ;REMOTE WRITE
FF1E: 4C ****                          JMP QRINIT          ;REMOTE INIT
FF21: 4C ****                          JMP QGWRITE         ;GRAPHIC WRITE
FF24:
FF24:                          ;---------------------------------------
FF24:                          ;
FF24:                          ; STRIP LOCAL RETURN ADDR,
FF24:                          ; STRIP PASCAL ADDR AND SAVE IN RETL,RETH
FF24:                          ; THEN RESTORE LOCAL RET ADDR & RETURN
FF24:                          ; MEANWHILE UNFOLD BIOS INTO DXXX
FF24:                          ;
FF24:                          ;---------------------------------------
FF24: 85 0C                    SAVERET     STA TT1             ;SAVE A REG
FF26: AD 83C0                              LDA 0C083           ;UNFOLD BIOS INTO DXXX
FF29: 68                                   PLA
FF2A: 85 0D                                STA TT2             ;LOCAL RET ADDR
FF2C: 68                                   PLA
FF2D: 85 0E                                STA TT3
FF2F: 68                                   PLA
FF30: 8D 1ABF                              STA RETL            ;PASCAL RETURN ADDR
FF33: 68                                   PLA
FF34: 8D 1BBF                              STA RETH
FF37: A5 0E                                LDA TT3
FF39: 48                                   PHA                 ;RESTORE LOCAL RET ADDR
FF3A: A5 0D                                LDA TT2
FF3C: 48                                   PHA
FF3D: A5 0C                    SKIPSAV     LDA TT1             ;RESTORE A REG
FF3F: 60                                   RTS                 ;BACK TO LOCAL CALLER
FF40:
FF40:                          ;---------------------------------------
FF40:                          ;
FF40:                          ; FOLD INTERP INTO DXXX
FF40:                          ; THEN RETURN TO PASCAL VIA
FF40:                          ; RETURN ADDR SAVED IN RETL,RETH
FF40:                          ;
FF40:                          ;---------------------------------------
FF40: 85 0C                    GOBACK      STA TT1             ;SAVE A REG
FF42: AD 1BBF                              LDA RETH
FF45: 48                                   PHA
FF46: AD 1ABF                              LDA RETL
FF49: 48                                   PHA                 ;PUT PASCAL RETURN ADDR ON STACK
FF4A: AD 8BC0                              LDA 0C08B           ;FOLD INTERP INTO DXXX
FF4D: A5 0C                    SKIPIT      LDA TT1             ;RESTORE A REG
FF4F: 60                                   RTS                 ;AND BACK TO PASCAL
FF50:
FF50:                          ;---------------------------------------
FF50:                          ;
FF50:                          ; PRESERVE OLD IORTS LOCATION
FF50:                          ;
FF50:                          ;---------------------------------------
FF50: 00 00 00 00 00 00 00                 .ORG 0FF58
FF58: 60                       IORTS       RTS                 ;FIXED RTS OP CODE
FF59:
```

```
FF59:                    ;-------------------------------------- --
FF59:                    ;
FF59:                    ; THIS CONCK VECTOR CALLED FROM DOS,
FF59:                    ; IF MOVED, CHANGE EQUATE IN DOS
FF59:                    ;
FF59:                    ;--------------------------------------
FF59: 4C 81D6                        JMP CONCK
FF5C:
FF5C:                    ;----------------------------------------
FF5C:                    ;
FF5C:                    ; THIS CONCK VECTOR USED BY KEYPRESS FUNCTION
FF5C:                    ; IF MOVED, CHANGE KEYPRESS FUNCTION
FF5C:                    ;
FF5C:                    ;----------------------------------------
FF5C: 20 24FF                        JSR SAVERET
FF5F: 20 81D6                        JSR CONCK
FF62: 4C 40FF                        JMP GOBACK
FF65:
FF65:                    ;----------------------------------------
FF65:                    ;
FF65:                    ; EACH BIOS ROUTINE UNFOLDS  DXXX RAM
FF65:                    ; THEN DOES ITS BUSINESS, THEN
FF65:                    ;   FOLDS RAM AGAIN
FF65:                    ;
FF65:                    ;----------------------------------------
FF65: AD 83C0            JSTART      LDA 0C083           ;WPROT,UNFOLD
FF68: 4C DDD5                        JMP START
FF6B: AD 83C0            JRESET      LDA 0C083
FF6E: 4C DCD5                        JMP RESET
FF71: AD 8BC0            JPASCAL     LDA 0C08B           ;FOLD IN INTERP
FF74: 4C 52D1                        JMP GOFORIT
FF01* 77FF
FF77: 20 24FF            GCREAD      JSR SAVERET
FF7A: 20 62D7                        JSR CREAD
FF7D: 4C 40FF                        JMP GOBACK
FF04* 80FF
FF80: 20 24FF            GCWRITE     JSR SAVERET
FF83: 20 DOD7                        JSR CWRITE
FF86: 4C 40FF                        JMP GOBACK
FF07* 89FF
FF89: 20 24FF            GCINIT      JSR SAVERET
FF8C: 20 34D7                        JSR CINIT
FF8F: 4C 40FF                        JMP GOBACK
FF0A* 92FF
FF92: 20 24FF            GPWRITE     JSR SAVERET
FF95: 20 30D8                        JSR PWRITE
FF98: 4C 40FF                        JMP GOBACK
FF0D* 9BFF
FF9B: 20 24FF            GPINIT      JSR SAVERET
FF9E: 20 88D7                        JSR PINIT
FFA1: 4C 40FF                        JMP GOBACK
FF10* A4FF
FFA4: 20 24FF            GDWRITE     JSR SAVERET
FFA7: 20 3CD0                        JSR DWRITE
FFAA: 4C 40FF                        JMP GOBACK
FF13* ADFF
```

```
FFAD: 20 24FF                GDREAD      JSR SAVERET
FFB0: 20 40D0                            JSR DREAD
FFB3: 4C 40FF                            JMP GOBACK
FF16* B6FF
FFB6: 20 24FF                GDINIT      JSR SAVERET
FFB9: 20 04D0                            JSR DINIT
FFBC: 4C 40FF                            JMP GOBACK
FF19* BFFF
FFBF: 20 24FF                GRREAD      JSR SAVERET
FFC2: 20 4ED8                            JSR RREAD
FFC5: 4C 40FF                            JMP GOBACK
FF1C* C8FF
FFC8: 20 24FF                GRWRITE     JSR SAVERET
FFCB: 20 09D8                            JSR RWRITE
FFCE: 4C 40FF                            JMP GOBACK
FF1F* D1FF
FFD1: 20 24FF                GRINIT      JSR SAVERET
FFD4: 20 9CD7                            JSR RINIT
FFD7: 4C 40FF                            JMP GOBACK
FF22* DAFF
FFDA: 20 24FF                GGWRITE     JSR SAVERET
FFDD: 20 58FF                            JSR IORTS          ;DO NOTHING FOR NOW
FFE0: 4C 40FF                            JMP GOBACK
FFE3:
FFE3:                        ;-----------------------------------------
FFE3:                        ;
FFE3:                        ; NMI AND RESET VECTORS
FFE3:                        ; FOR NOW, NO INTERRUPTS.  MAYBE SOON
FFE3:                        ;
FFE3:                        ;-----------------------------------------
FFE3: 00 00 00 00 00 00 00               .ORG OFFF6
FFF6: 0100                               .WORD  1              ;RELEASE VERSION 1
FFF8: 65FF                               .WORD  JSTART         ;START POINT FROM BOOT
FFFA: 6BFF                               .WORD  JRESET         ;NMI VECTOR
FFFC: 6BFF                               .WORD  JRESET         ;RESET VECTOR
FFFE: 6BFF                               .WORD  JRESET         ;IRG VECTOR
0000:
0000:                        ;-------------------------------------
0000:                        ;
0000:                        ; THAT'S ALL FOLKS...
0000:                        ;
0000:                        ;-------------------------------------
0000:
0000:
0000:
0000:
0000:
0000:
0000:
0000:
0000:                                .END
```

AB - Absolute LB - Label UD - Undefined MC - Macro
RF - Ref DF - Def PR - Proc FC - Func
PB - Public PV - Private CS - Consts

```
ADJLFT  LB DABB  ADJOK   LB DAA1  ADJRT   LB DAAC  ADJRTS  LB DAAB  ADJUST  LB DA92  ADVANCE LB D9B6  ADVRTS  LB D9B
BAS1H   AB 00F1  BAS1L   AB 00F0  BAS2H   AB 00F3  BAS2L   AB 00F2  BASCAL  LB D9F1  BELL    LB D957  BELL2   LB D95
BIOS    PR ----  BMPRTS  LB D733  BREAK   AB BF16  BSCL2   LB DA02  BSRET   LB D986  BUFFER  AB 0200  BUFOK   LB D71
BUMP    LB D72C  BXS1H   AB 0005  BXS1L   AB 0004  BXS2H   AB 0007  BXS2L   AB 0006  CBUFLEN AB 004E  CH      AB 00F
CHECKH  AB 000B  CHECKL  AB 000A  CINIT   LB D734  CINIT2  LB D75F  CKCTRL  LB D8F3  CKEXIT  LB D725  CKNX    LB D66
CKPAGE  LB D667  CKPTRH  AB 0009  CKPTRL  AB 0008  CLEOL   LB D9A7  CLEOS   LB D989  CLNXL   LB D98F  CLOOP   LB D9A
CLRIO   LB D7F4  CLRIO1  LB D7AD  CLRIO2  LB D81C  CLRIO3  LB D786  CNO5BYTS LB D60D  CNO7BYTS LB D611  CNTRL   LB D8F
COMCK   LB D6CA  CONBUF  AB 03B1  CONCK   LB D681  CONFLGS AB BF15  COPY    LB DA14  COPY1   LB D932  COPY2   LB D93
CR      LB D918  CREAD   LB D762  CREAD2  LB D771  CTRLCH  LB D87B  CTRLJMP LB D887  CV      AB 00F5  CWRITE  LB D7D
DINIT   AB D004  DOCTRL  LB D8FC  DOIT    LB DA8B  DONECK  LB D71D  DREAD   AB D040  DRESET  AB D017  DWRITE  AB D03
ESCAPE  LB D90C  ESCNT   AB BF12  FARRT   LB DAB8  FIRST   AB 00F0  FORM    LB D97A  GENIT   LB D791  GENR    LB D85
GENW    LB D83F  GENW2   LB D843  GOBACK  LB FF40  GOFORIT AB D152  GOTOXY  LB D912  HIDNLP  LB DA5E  HOME    LB D98
HSHIFT  LB DA23  HTAB    LB DA7E  ICOM    LB D7A3  INVERT  LB D9EC  IORTS   LB FF58  ISER    LB D7B0  JDONCK  LB D6E
JINV    LB D979  JPASCAL LB FF71  JRESET  LB FF6B  JRET    LB D969  JSTART  LB FF65  JUMP1   AB 0002  JUMP2   AB 000
LF      LB D91D  LF2     LB D923  LFFLAG  AB BF0F  LFPASS  LB D83A  LOP1    LB DA38  LOP2    LB DA3D  NCTRLS  AB 000
NDBS    LB D973  NLEFT   AB BF11  NOCRY   LB D671  NOPROM  LB D634  NOTBRK  LB D6F6  NOTFLUS LB D706  NOTFOL  LB D6D
NOTK    LB D6AA  NOTSEG  LB D8DC  NOTSTOP LB D6E6  NOWRP1  LB DA54  NOWRP2  LB DA63  NTTAB   LB D6BC  NXTCRD  LB D5F
NXTYP   LB D617  OFFLFT  LB D9D8  OFFLINE LB D84B  OFFRT   LB D9DE  OKDELT  LB DA33  OOPSX   LB D8B5  OOPSY   LB D8C
PINIT   LB D788  PWRITE  LB D830  QCINIT  LB FF89  QCREAD  LB FF77  QCWRITE LB FF80  QDINIT  LB FFB6  QDREAD  LB FFA
QDWRITE LB FFA4  QQWRITE LB FFDA  QPINIT  LB FF9D  QPWRITE LB FF92  QRINIT  LB FFD1  QRREAD  LB FFBF  QRWRITE LB FFC
RANDH   AB BF14  RANDL   AB BF13  RCOM    LB D85D  RESET   LB D5DC  RETH    AB BF1B  RETL    AB BF1A  REVLF   LB D96
RINIT   LB D79C  RNDINC  LB D687  RNDOK   LB D68F  RPTR    AB BF18  RREAD   LB D84E  RSER    LB D86C  RTS7    LB DA2
RWRITE  LB D809  SAVERET LB FF24  SCRMODE AB BF0E  SCROLL  LB D928  SEGRET  LB D8DB  SER1    LB D7B9  SETY    LB D8B
SKIPIORT LB D615 SKIPIT  LB FF4D  SKIPSAV LB FF3D  SKPCOUT LB D948  SLTTYPS AB BFF8  START   LB D5DD  STARTUP LB D66
STOADV  LB D9B3  STOR    LB D62E  STOR2   LB D9E1  STORE   LB D9C0  STORE1  LB D9D5  STORE2  LB D9E9  SYSCOM  AB 00F
TAB2    LB DA85  TAB3    LB DA89  TEMP1   AB 00F6  TEMP2   AB 00F7  TRYNXT  LB D629  TSTKBD  LB D69A  TT1     AB 000
TT2     AB 000D  TT3     AB 000E  UPPER   LB D8EA  VIDOUT  LB D7E7  VISLP   LB DA4F  VOUT2   LB D89F  WAIT    LB D67
WAIT2   LB D676  WAIT3   LB D677  WCOM    LB D81F  WPRN    LB D811  WPTR    AB BF19  WSER    LB D7F7  XOK     LB D8B
YOK     LB D8C9  ZERLP   LB D5E5  ZEROH   AB 0001  ZEROL   AB 0000
```

```
Current minimum space is 7938 words
D64E* ECD9
D6AF* 7EDA
D75D* 89DA
D763* 92DA
D665* 71FF

Assembly complete:       1202 lines
     0   Errors flagged on this Assembly
```

APPLICATIONS NOTE FOR PASCAL

PASCAL LONG INTEGER FIX

An error in the implementation of long integers in the Apple PASCAL language system results in a stack crash during a compare operation.

This program is designed to repair the library module LONGINTEGER. To use, type the program in, then compile and execute. All libraries should be updated with this program. Save the text for possible future use.

```
{$I-}
PROGRAM FIXCOMPARE;

TYPE  DISKINFO = RECORD
                   DADDR:INTEGER;
                   LENG: INTEGER
                 END;

      NAME = PACKED ARRAY[1..8] OF CHAR;

      SEGDIC = RECORD
                 DINFO: ARRAY[0..15] OF DISKINFO;
                 SEGNAME:ARRAY[0..15] OF NAME;
                 FILLER: ARRAY[1..416] OF INTEGER
               END;

      BLOCK = PACKED ARRAY[0..511] OF 0..255;

VAR   I,J:INTEGER;
      NOTFOUND:BOOLEAN;
      F:FILE;
      S:STRING;
      BLOCKZERO:SEGDIC;
      DATA:BLOCK;

PROCEDURE READERROR;
BEGIN
  WRITE('BAD BLOCK IN LIBRARY'); EXIT(PROGRAM)
END;

BEGIN
  NOTFOUND:=TRUE;
  REPEAT
    WRITE('NAME OF LIBRARY FILE:'); READLN(S);
    RESET(F,S);
  UNTIL EOF OR (IORESULT=0);
  IF BLOCKREAD(F,BLOCKZERO,1,0) <> 1 THEN READERROR;
  FOR I := 0 TO 15 DO
    BEGIN
      IF BLOCKZERO.SEGNAME[I] = 'LONGINTI' THEN
        BEGIN
          NOTFOUND := FALSE;
          J := BLOCKZERO.DINFO[I].DADDR + 1;
          IF BLOCKREAD(F,DATA,1,J)<>1 THEN READERROR;
          IF DATA[495]=244 THEN
            IF DATA[494]=208 THEN
              BEGIN
                WRITELN('LONG INTEGER PATCH BEING MADE');
                DATA[494]:=240;
                IF BLOCKWRITE(F,DATA,1,J) = 1 THEN
                  WRITELN('PATCH COMPLETE')
                ELSE
                  WRITELN('ERROR WHILE WRITING PATCH, SEGMENT ',I);
              END
```

```
          ELSE
            IF DATA[494]=240 THEN
              WRITELN('SEGMENT ',I,
                      ' - LONG INTEGER UNIT HAS ALREADY BEEN FIXED')
            ELSE
              WRITELN('CAN''T RECOGNIZE LONG INTEGER UNIT IN SEGMENT ',I)
        ELSE
          WRITELN('CAN''T RECOGNIZE LONG INTEGER UNIT IN SEGMENT ',I)
      END
    END;
  IF NOTFOUND THEN WRITELN('NO LONG INTEGER UNIT FOUND')
  ELSE WRITELN('PROGRAM COMPLETE');
END.
```

PASCAL HI-RES LOAD/SAVE TO DISK

This demo program creates a hi-res picture in PASCAL, then saves it to disk. It is then reloaded and displayed.

The "Uses Turtlegraphics" statement allocates space for the hi-res screen, and should be referenced even if Turtlegraphics are not actually used. Note that the "Close (F,Lock)" closes the file and places it permanently into the volume directory.

```
PROGRAM DEMOPIC;
(*                                                    *)
(*     PROGRAM LOADS AND SAVES HIRES SCREEN TO DISK    *)
(*    APPLE COMPUTER 12/79  BY JO & CHARLIE KELLNER    *)
(*       BASED ON "SLIDE SHOW" BY BILL ATKINSON        *)
(*                                                    *)
USES TURTLEGRAPHICS, APPLESTUFF;
CONST HIRESP1 = 8192;
VAR CHEAT:  RECORD  CASE BOOLEAN OF
                TRUE: (INTPART: INTEGER);
                FALSE: (PTRPART:^INTEGER);
            END;

PROCEDURE DRAWPICS;     (* THIS CAN BE REPLACED WITH ANY GRAPHICS ROUTINE *)
BEGIN
  MOVETO (0,0);  PENCOLOR (WHITE);
  MOVETO (279,0);  TURN (90);
  MOVETO (279,191);  TURN (90);
  MOVETO (0,191);  TURN (90);
  MOVETO (0,0);  PENCOLOR (NONE);
  MOVETO (75,95);  WSTRING (' THIS IS A TEST ');
  MOVETO (28,5);  WSTRING ('< PRESS RETURN TO EXIT PROGRAM >')
END;

PROCEDURE BLOAD (FILENAME: STRING);
VAR IO: INTEGER;
    F: FILE;
BEGIN
  CHEAT.INTPART:=HIRESP1;
  RESET(F,FILENAME);
  IO:=BLOCKREAD(F,CHEAT.PTRPART^,16);
  CLOSE(F,LOCK)
END;

PROCEDURE BSAVE (FILENAME:STRING);
VAR IO: INTEGER;
    F: FILE;
BEGIN
  CHEAT.INTPART:=HIRESP1;
  REWRITE (F,FILENAME);
  IO:=BLOCKWRITE (F,CHEAT.PTRPART^,16);
  CLOSE (F,LOCK)
END;

BEGIN   (* MAIN PROGRAM *)
INITTURTLE;
DRAWPICS;
BSAVE (':DEMO.PIC');
FILLSCREEN (BLACK);       (* CLEARS HIRES SCREEN *)
BLOAD (':DEMO.PIC');
REPEAT UNTIL KEYPRESS;
TEXTMODE
END.
```

EXTERNAL TERMINAL SETUP IN PASCAL

Pascal supports a variety of external terminals which can be used to display 80 character lines and upper/lower case. Pages 240-252 of the Apple PASCAL Reference manual describe the procedure for reconfiguring the Pascal system to communicate with the external terminal. You must bind the appropriate GOTO routine into SYSTEM.PASCAL using APPLE3:BINDER and use APPLE3:SETUP to tailor SYSTEM.MISCINFO for the specific terminal parameters.

EXTERNAL TERMINALS WITH APPLE SERIAL CARD

The terminal may be connected to the Pascal system using the High-Speed Serial Card in place of the Communications Card. If the serial card is used, the eighth data bit (most significant bit) switch in the terminal must be set to 0. Refer to the operating manual or contact the manufacturer for the location of this switch. With the serial card, all type-ahead is eliminated.

SOROC IQ120

GOTO CODE: use APPLE3:SOROC.GOTO No modifications are necessary.

SETUP PARAMETERS:			PREFIXED:	
Backspace	CTRL-H	(08)	[Delete Character]	FALSE
Editor Accept Key	CTRL-C	(03)	[Editor Accept Key]	FALSE
Editor Escape Key	ESC	(27)	[Editor Escape Key]	FALSE
Erase Line	NUL	(00)	[Erase Line]	FALSE
Erase Screen	"*"	(42)	[Erase Screen]	TRUE
Erase to End of Line	"T"	(84)	[Erase to End of Line]	TRUE
Erase to End of Screen	"Y"	(89)	[Erase to End of Screen]	TRUE
Has 8510A	FALSE		[Key for Break]	FALSE
Has Clock	FALSE		[Key for Flush]	FALSE
Has Lower Case	TRUE		[Key for Moving Cursor Down]	FALSE
Has Random Cursor Addr.	TRUE		[Key for Moving Cursor Left]	FALSE
Has Slow Terminal	FALSE		[Key for Moving Cursor Right]	FALSE
Key for Break	NUL	(00)	[Key for Moving Cursor Up]	FALSE
Key for Flush	CTRL-F	(06)	[Key for Stop]	FALSE
Key for Stop	CTRL-S	(19)	[Key to Delete Character]	FALSE
Key to Delete Character	CTRL-H	(08)	[Key to Delete Line]	FALSE
Key to Delete Line	DEL	(127)	[Key to End File]	FALSE
Key to End File	CTRL-C	(03)	[Move Cursor Home]	FALSE
*Key to Move Cursor Down	CTRL-J	(10)	[Move Cursor Right]	FALSE
*Key to Move Cursor Left	CTRL-H	(08)	[Move Cursor Up]	FALSE
*Key to Move Cursor Right	CTRL-L	(12)	[Non-printing Character]	FALSE
*Key to Move Cursor Up	CTRL-K	(11)	Screen Height	24
Lead-in from Keyboard	NUL	(00)	Screen Width	80
Lead-in to Screen	ESC	(27)	Student	FALSE
Move Cursor Home	CTRL-^	(30)	Vertical Move Delay	0
Move Cursor Right	CTRL-L	(12)		
Move Cursor Up	CTRL-K	(11)	* These may be altered by the user for convenience.	
Non-printing Character	"?"	(63)		

HAZELTINE 1500

GOTO CODE: use APPLE3:HAZEL.GOTO No modifications are necessary.

SETUP PARAMETERS:		PREFIXED:	
Backspace	CTRL-H (08)	[Delete Character]	FALSE
Editor Accept Key	CTRL-C (03)	[Editor Accept Key]	FALSE
Editor Escape Key	ESC (27)	[Editor Escape Key]	FALSE
Erase Line	NUL (00)	[Erase Line]	FALSE
Erase Screen	CTRL-\ (28)	[Erase Screen]	TRUE
Erase to End of Line	CTRL-O (15)	[Erase to End of Line]	TRUE
Erase to End of Screen	CTRL-X (24)	[Erase to End of Screen]	TRUE
Has 8510A	FALSE	[Key for Break]	FALSE
Has Clock	FALSE	[Key for Flush]	FALSE
Has Lower Case	TRUE	[Key for Moving Cursor Down]	FALSE
Has Random Cursor Addr.	TRUE	[Key for Moving Cursor Left]	FALSE
Has Slow Terminal	FALSE	[Key for Moving Cursor Right]	FALSE
Key for Break	NUL (00)	[Key for Moving Cursor Up]	FALSE
Key for Flush	CTRL-F (06)	[Key for Stop]	FALSE
Key for Stop	CTRL-S (19)	[Key to Delete Character]	FALSE
Key to Delete Character	CTRL-H (08)	[Key to Delete Line]	FALSE
Key to Delete Line	DEL (127)	[Key to End File]	FALSE
Key to End File	CTRL-C (03)	[Move Cursor Home]	TRUE
*Key to Move Cursor Down	CTRL-K (11)	[Move Cursor Right]	FALSE
*Key to Move Cursor Left	CTRL-H (08)	[Move Cursor Up]	TRUE
*Key to Move Cursor Right	CTRL-P (16)	[Non-printing Character]	FALSE
*Key to Move Cursor Up	CTRL-L (12)	Screen Height	24
Lead-in from Keyboard	NUL (00)	Screen Width	80
Lead-in to Screen	"~" (126)	Student	FALSE
Move Cursor Home	CTRL-R (18)	Vertical Move Delay	0
Move Cursor Right	CTRL-P (16)		
Move Cursor Up	CTRL-L (12)	* These may be altered by the user for convenience.	
Non-printing Character	"?" (63)		

ADDITIONAL NOTES: The four parity switches must be set to OFF. The CR/Auto LF switch must be set to CR. These DIP switches are located beneath the front panel plate of the terminal.

HAZELTINE 1510

GOTO CODE: use APPLE3:HAZEL.GOTO No modifications are necessary.

SETUP PARAMETERS:	
Backspace	CTRL-H (08)
Editor Accept Key	CTRL-C (03)
Editor Escape Key	ESC (27)
Erase Line	NUL (00)
Erase Screen	CTRL-\ (28)
Erase to End of Line	CTRL-O (15)
Erase to End of Screen	CTRL-X (24)
Has 8510A	FALSE
Has Clock	FALSE
Has Lower Case	TRUE
Has Random Cursor Addr.	TRUE
Has Slow Terminal	FALSE
Key for Break	NUL (00)
Key for Flush	CTRL-F (06)
Key for Stop	CTRL-S (19)
Key to Delete Character	CTRL-H (08)
Key to Delete Line	DEL (127)
Key to End File	CTRL-C (03)
*Key to Move Cursor Down	CTRL-K (11)
*Key to Move Cursor Left	CTRL-H (08)
*Key to Move Cursor Right	CTRL-P (16)
*Key to Move Cursor Up	CTRL-L (12)
Lead-in from Keyboard	"~" (126)
Lead-in to Screen	"~" (126)
Move Cursor Home	CTRL-R (18)
Move Cursor Right	CTRL-P (16)
Move Cursor Up	CTRL-L (12)
Non-printing Character	"?" (63)

PREFIXED:	
[Delete Character]	FALSE
[Editor Accept Key]	FALSE
[Editor Escape Key]	FALSE
[Erase Line]	FALSE
[Erase Screen]	TRUE
[Erase to End of Line]	TRUE
[Erase to End of Screen]	TRUE
[Key for Break]	FALSE
[Key for Flush]	FALSE
[Key for Moving Cursor Down]	TRUE
[Key for Moving Cursor Left]	FALSE
[Key for Moving Cursor Right]	FALSE
[Key for Moving Cursor Up]	TRUE
[Key for Stop]	FALSE
[Key to Delete Character]	FALSE
[Key to Delete Line]	FALSE
[Key to End File]	FALSE
[Move Cursor Home]	TRUE
[Move Cursor Right]	FALSE
[Move Cursor Up]	TRUE
[Non-printing Character]	FALSE
Screen Height	24
Screen Width	80
Student	FALSE
Vertical Move Delay	0

* These may be altered by the user for convenience.

PASCAL SEGMENT PROCEDURES

Modular programming means the separation of procedures and functions, or groups of them, from the main program. Modularization may be achieved at the P-code level by designating a procedure, function, or group of procedures and functions as a SEGMENT. The result is that its code is not loaded into memory until it is called by some other part of the program. As soon as the SEGMENT procedure or function is no longer active it is "swapped out"; that is, its memory space is made available for some other use. Since the Pascal system does not support a "chain" command, segmenting provides a way of simulating chaining.

Segmented programs can be compiled as one file, or can be broken into individual segments and compiled separately. The demo programs in this note are used to illustrate how segment procedures can be compiled separately and merged using the PASCAL Librarian function.

CREATING A SEGMENTED PROGRAM FROM SEPARATELY COMPILED FILES

Each segment, plus the main program, must be contained in a program framework in which ALL declaration sections are identical from program to program (see examples), but only one segment procedure in each program contains active code. This "placeholding" is required since each code segment must be numbered uniquely.

Compile each program separately. Execute APPLE3:LIBRARY and follow the directions as if creating a library unit. (See pages 235-239 in the white reference manual for instructions on using the Librarian.) Note that with each new link code file, all segments are listed, but only one contains code: the others are the "placeholders". Transfer the active segment (only) to the output code file. Repeat for each segment and for the mainline code. Your segmented program is now ready for execution.

IMPORTANT NOTE: The output code file MUST be specified as <filename>.CODE. Omitting the .CODE suffix will result in a non-executable datafile.

PROGRAM LISTINGS

NOTE: These programs contain no error-checking.

```
PROGRAM Segmain;   (* Framework for mainline code *)

VAR I:Integer;

PROCEDURE Print(S:String);Forward;

SEGMENT PROCEDURE One;
Begin
End;

SEGMENT PROCEDURE Two;
Begin
End;

SEGMENT PROCEDURE Three;
Begin
End;

PROCEDURE Print;
Begin
  Writeln(S);Writeln;
End;

BEGIN  (* Main *)
Repeat
  Write('Enter Integer 1-3, 0 to Quit : ');Readln(I);
  Case I of
   1: One;
   2: Two;
   3: Three;
   0: Begin
       Print('The End');
       Exit(Program);
      End;
  End;
Until False;
END.
```

```
PROGRAM Seg1;    (* Framework for Segment One *)

VAR I:Integer;

PROCEDURE Print(S:String);Forward;

SEGMENT PROCEDURE One;
Begin
  Print('Segment One');
End;

SEGMENT PROCEDURE Two;
Begin
End;

SEGMENT PROCEDURE Three;
Begin
End;

PROCEDURE Print;
Begin
End;

BEGIN  (* Main *)
END.
```

```
PROGRAM Seg2;     (* Framework for Segment Two *)

VAR I:Integer;

PROCEDURE Print(S:String);Forward;

SEGMENT PROCEDURE One;
Begin
End;

SEGMENT PROCEDURE Two;
Begin
  Print('Segment Two');
End;

SEGMENT PROCEDURE Three;
Begin
End;

PROCEDURE Print;
Begin
End;

BEGIN  (* Main *)
END.
```

```
PROGRAM Seg3;     (* Framework for Segment Three *)

VAR I:Integer;

PROCEDURE Print(S:String);Forward;

SEGMENT PROCEDURE One;
Begin
End;

SEGMENT PROCEDURE Two;
Begin
End;

SEGMENT PROCEDURE Three;
Begin
  Print('Segment Three');
End;

PROCEDURE Print;
Begin
End;

BEGIN  (* Main *)
END.
```

PASCAL UNITS

Modular programming means the separation of procedures and functions, or groups of them, from the main program. Source language modules are called UNITs, and are incorporated in libraries for use with Pascal programs. Units may consist of procedures, functions, or a combination of these, in Pascal and in assembly language.

Separate compilation has several advantages in the development of any program, because it allows you to approach the task as a group of smaller tasks which are linked together in a logical manner. The host program must contain a USES statement to utilize routines from the UNIT.

There are two principal kinds of UNITs: Regular UNITs, and Intrinsic UNITs. When a host program USES a Regular UNIT, the UNIT's code is inserted into the host program's codefile by the Linker. This needs to be done only once unless the UNIT is modified and recompiled; then it must be relinked into the host program.

When a host program USES an Intrinsic UNIT, the UNIT's code remains in the library file and is automatically loaded into memory when the host program is executed. This keeps the size of the host program's codefile down; it also allows the UNIT to be modified and recompiled without the need to relink. If the UNIT resides in the SYSTEM.LIBRARY, the Linker will be called automatically. Otherwise, you must explicitly invoke the Linker.

Pages 187-195 in the Pascal Reference Manual explain the syntax and structure of UNITs.

Separate UNITs do not work in the current Pascal implementation.

Yes! There Is a Fix for APPEND in DOS 3.2 (and 3.2.1)!

The problem with APPEND in DOS 3.2 is that DOS doesn't write an End Of File marker on the disk when you close a file. DOS normally fills new sectors with EOF markers, so the newly APPENDed information usually has an EOF after the last character. However, when the last character of the file falls exactly at the end of a sector, DOS doesn't find a new sector to fill with EOF markers. The next time DOS does an APPEND it can't find the EOF marker and defaults back to the beginning of the file.

The fix is to write out an EOF marker before closing the file after each write. Here is a five byte routine that will supply an EOF. It can be moved to any address if you are already using 768 to 772.

```
10  LET D$ = CHR$( 4)
20  POKE 768,169
30  POKE 769,0
40  POKE 770,76
50  POKE 771,237
60  POKE 772,253
70  REM  NOW TO USE IT!
80  PRINT D$;"APPEND FILE"
90  PRINT D$;"WRITE FILE"
100  PRINT "THIS IS DATA"
110  PRINT "SO IS THIS"
120  CALL 768: PRINT : REM THIS IS IT!
130  PRINT D$;"CLOSE FILE"
140  END
```

Using this method, one need never worry about APPEND overwriting the start of a file.

apple computer inc.
10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010